University of Nevada, Reno

A Framework for a Sign Language Interfacing System

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering.

by

Beifang Yi

Dr. Frederick C. Harris, Jr., Dissertation advisor

May 2006

UNIVERSITY
OF NEVADA
RENO

THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

BEIFANG YI

entitled

A Framework for a Sign Language Interfacing System

be accepted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

[illegible]
Frederick C. Harris, Jr., Ph.D. , Advisor

[illegible]
Sergiu M. Dascalu, Ph.D. , Committee Member

Carl G. Looney
Carl G. Looney, Ph.D. , Committee Member

[illegible]
George Danko, Ph.D. , Committee Member

Thomas Nickles
Thomas Nickles, Ph.D. , Graduate School Representative

[illegible]
Marsha H. Read, Ph. D., Associate Dean, Graduate School

May, 2006

Abstract

Sign languages have been proven to be natural languages, as capable of expressing human thoughts and emotions as traditional (spoken) languages are. The distinct visual and spatial nature of sign languages makes it difficult to develop an interfacing system as a communication medium platform for sign language users. This dissertation targets this problem by presenting some explorations in the areas of computer graphics, interface design, and human-computer interactions with emphasis on software development and implementation. First, a virtual human body has been constructed as a set of functional body components with the consideration of the biomechanics of the human body. Then based on the virtual body, virtual gestures are created by controlling the movements of the functional components, with substantial concentration on the creation of natural hand configurations and the application of body joint motion constraints. Finally, from the virtual gestures, sign language linguistic parts can be constructed by using the Movement-Hold Model. An ease-of-use graphical user interface is designed and implemented that wraps up of all the operations of gesture generation and editing, gesture database management, and creation, editing, storage, and retrieval of sign language linguistic parts.

In this dissertation, we propose a sign language interfacing system with verification of preliminary results that differs from other computer-related sign language systems in its (1) flexibility of creating any sign language linguistic parts (from phonemes to sentences), (2) "openness" of application to any sign languages, and (3) capability of being used as a sign language "editor" like a word processor for spoken languages.

Dedication

This work is dedicated to three great women from three generations in my life:

- the youngest one, Fandi, my daughter, whose joking about my speaking English with Chinese accent nearly slowed down my writing of this dissertation.

- the middle one, Zhifeng, my wife, who gave up a well-paid job in China and joined me here in America. She has allowed me to concentrate on my PhD work.

- the oldest, Xueying, my mother, who is unable to read any Chinese words (her native language), even her own names. She has always been an inspiration throughout my persistent academic explorations: whenever I am trying to find an excuse of staying away from a book or my academic work, I immediately remember those terrible days when she had to labor on the farmland, ten hours a day, without any weekends or vacations, and half of her month's payment could not afford a cartoon book which I had dreamed of almost every night. I wish that this dissertation, written in a language totally foreign to her, would be a precious gift for her.

Acknowledgments

I would like to thank Dr. Harris, my advisor, for his generous help when it was most needed and for his guidance, support, and encouragement during my PhD years.

I am very grateful to Dr. Danko, Dr. Dascalu, Dr. Looney, and Dr. Nickles for serving on my committee and for giving of their valuable time. Dr. Danko's *Robotics* course, Dr. Dascalu's *Human-Computer Interaction* course, Dr. Harris's *Computer Graphics* course, and Dr. Looney's *Information Fusion* course have been a big help in my academic studies. Dr. Nickles' *Philosophy of Mind* course gave me some wonderful inspirations, some subconsciously, that led me to research a topic that is not exclusively in the area of "engineering."

I am also obliged to Dr. Bebis for his kind support of my work on modeling the human hand. The regular meetings with him, together with Dr. Harris, are full of wonderful memories.

Moreover, I am indebted to Mrs. Harris for her careful and conscientious proofreading of my papers, thesis, dissertation, and even some of my presentation slides. Her generous help has been a benefit for me to improve my English writing skills.

Mr. Steven Arnold's participation in the project has saved me much time in providing a workable demo; Mr. Lance Hutchinson's prompt action and ready assistance when the lab machines broke down has made the the project go smoothly.

Last but not least, special thanks should be poured out to my parents, my daughter, and my wife, for their understanding of and patience with my career change from an engineer to a student, and to my daughter and my wife who, giving up Chinese middle-class comfortable life, have accompanied me during my PhD years and led with me a hard student life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When hearing the word *language*, we automatically think of *words*, *phrases*, and *sentences*. In today's information age, a text editor is an indispensable language tool, and text editors with standardized text fonts have greatly enriched the communications between people speaking different languages. By exploiting our oral and aural faculties, we talk, listen, write, and read; we use the different representative patterns of that text format to communicate with each other, exchanging information and expressing our thoughts and emotions. But we often forget that there are other channels which can be used to achieve the same functions as the traditional (spoken) languages do.

When we speak, we make gestures, especially when we want to express strong feelings such as anger, distress, eagerness, and ecstasy. Most often, however, we gesture without our awareness of doing so. Hard hearing people rely exclusively on gesturing to accomplish what hearing people can do with both speaking and gesturing. Studies on languages and gestures suggest that both spoken languages and gestures are only two different representations of a single system in the human mind [66, 67]. Fundamentally, languages and gestures differ only in their modalities of production (oral *vs.* gestural) and perception (aural *vs.* visual).

Naturally, people with hearing impairments depend on visual and gestural faculties as their main avenues of communication. Sign languages have been proven linguistically to be natural languages [59, 94], just as capable of expressing human thoughts and feelings as traditional languages are. Sign languages are also a medium between the hearing world and the deaf community.

There are some notations used to transcribe the signing information [7, 14, 25, 93], but there exists no standard writing system that can be used for all the sign languages. In fact, there may be different notation systems even in the same sign language [50]. Furthermore, to hard of hearing people a writing system of a sign language is like a second language to them because their native language depends heavily on visual and gestural channels and the physical space around them.

The visual and spatial nature of sign languages contributes to the lack of "editors" in such languages. The current writing systems, while making full use of various suggestive 2D icons or phonetic symbols, are indirect, unnatural transcriptions and transformations of the 3D expressions inherent in sign languages. This symbol representation for a sign language is, in fact, like a text encoding of spatial contents. There are three options for non-text representations and displays: (1) analog (video recording); (2) digital (recording on a compact disk); and (3) parametric (virtual signing gestures generated at runtime from gesture feature parameters). Due to their inherent limitations, the first two display formats are unsuitable for smooth signing: they are pre-recorded and the video clips cannot be edited and combined into a larger and smoother video clip. A promising, direct and natural writing system representation is offered by the third option—a human avatar signing in a virtual environment.

The use of virtual human figures in sign language studies is ongoing [3, 22, 50]. The two main problems with these studies' results are that the users have to know

English to utilize them and that there is no user control over the signing avatar. Therefore, the users cannot create new sign language “words” or “phrases,” which is a significant limitation because a natural language should be open for extending its contents. The lack of easy-to-use interfaces also makes it difficult for beginners to benefit from the results of these studies.

As an exploration to solve these problems, we have done work in the areas of computer graphics and human-computer interactions (HCI) with the emphasis on human body modeling, user interface design, and software implementation, aiming to build up a framework of a sign language interfacing system. The system provides a convenient platform to construct, manage, and edit virtual gestures. From these gestures, meaningful signing parts can be built. Design guidelines and specifications on creating, storing, retrieving, and editing linguistic parts along with some results have been proposed.

The distinctive features of this sign language interfacing system that differ from other computer-related sign language projects lie in its flexibility, “openness”, and capability of being a potential sign language “editor”. First, any sign language linguistic parts, from phonemes to sentences, can be created with the system. These linguistic parts, once built, may be stored to and retrieved from sign language databases, allowing for editing and construction of other or larger linguistic parts. Names for and keywords related with the created linguistic parts can be assigned by users and saved into the databases; thus users can quickly find interested signs through the names or keywords. User-friendly and efficient graphical user interfaces make it possible and effective to create, edit, store, and retrieve any signs in a short time.

The “openness” of the system means that the interfacing system can used for any sign language and creation of vocabulary for a chosen sign language. The system has been developed under Linux operating system with Open Source software packages capable of *Internationalization* features; thus the system can be ported to any other environments. Because users have full control over the avatar in the system, creation of new words and phrases for a sign language is one of the basic functions of the system, as discussed previously.

The ultimate goal of this interfacing system is to create a sign language "editing" system, like word processors for the traditional languages. We have finished the first step toward this goal: (1) retrieval of sign language words from the database, (2) combination of these words into a sentence, and (3) editing the retrieved words (signing gestures) according sign language grammar. This first step has solved the technical problems of creating a sign language "editor" and provided a basic and necessary working platform to enhance the current system in practical application with a selected sign language.

The organization of this dissertation is as follows: Chapter 2 provides the background in and basic knowledge of the HCI, the biomechanics and modeling of the human body, and sign languages. Chapter 3 gives an overview of the proposed sign language interfacing system—its design philosophy and functionality. Chapter 4 is concerned with human body modeling with the concentration on hand modeling and the application of hand constraints. Results will also be shown in this chapter. Chapter 5 presents the design, creation, and management of virtual gestures by categorizing the virtual gestures and implementing gesture data structures. Some initial results will be provided. Chapter 6 discusses the creation of sign language linguistic parts such as phonological and morphological parts. Preliminary results and implementation prototypes will be given as well as a proposal of how to "write" in sign language. Chapter 7 presents conclusions and future work.

Chapter 2

Review of the Related Literature

The development of a sign language interfacing system is closely related to the advancement of computer technologies and their applications in the studies of sign languages: modeling of the human body, design of virtual gestures, construction of virtual gesture sessions, and conversion of the sessions into virtual signings in a particular sign language. All of these topics can be logically correlated within the scope of human-computer interaction. In this chapter, some of the recent work related with the proposed sign language interfacing system will be introduced: human-computer interaction, modeling of the human body, and sign language studies.

2.1 Human-Computer Interaction

Human-computer interaction/interface (HCI), a relatively new but vigorously developing discipline, has not only evolved as an active research area with the rapid development of computer science and engineering technologies but has also broadened its scope into other subjects such as psychology, sociology, anthropology, and industrial design. During the past few years, a significant number of academic institutions and industrial corporations have paid more and more attention to HCI and its study of the ease-of-use environment between man and machine. Many various research topics have employed the concepts of HCI, and more and more industrial products that have no relation to one another are making use of the methods of HCI. Different researchers and users have different views of HCI and even different definitions of HCI. In this section we first introduce one or two definitions representative of HCI and the research concerned with HCI and then discuss some important HCI issues.

2.1.1 HCI Definition

To define a scientific discipline, it always is a good idea first to look at which research areas the discipline has emphasized. As for HCI, Andrew Sears in [82] presented three definitions by other scholars for HCI: (1) "Human-Computer Interaction (HCI) is about designing computer systems that support people so that they can carry out their activities productively and safely"; (2) "Human-Computer Interaction (or HCI) is, put simply, the study of people, computer technology and the ways these influence each other. We study HCI to determine how we can make this computer technology more usable by people"; and (3) "Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." From the above definitions, four important ideas stand out: people, computing systems, interaction, and usability.

The term *people* here refers to an individual or a group of people working together to solve a problem [82]. Although many people think of a *user*-centered interface when considering HCI, Miller [69] prefers the term *human*-computer interface instead of *user* interface because the domain of the user and the user population have been changing rapidly. Here *people* and *human* mean the same. *Computing systems* differ from traditional *computers*. Computing systems include computer vision, computer graphics, image processing, operating system, and informatics. Usability is concerned with a system's simplicity to learn and its efficiency and effectiveness to use. Researchers evaluate the usability of a system they have developed by implementing and testing the design. Finally, *interaction or interface* is the key word in HCI. Hewett *et al.* [47] describe HCI as an interdisciplinary area with emphasis on the "joint performance of tasks by humans and machines," and "the structure of communication between human and machines." For example, in the domain of computer science, the *interaction* takes the form of "application design and engineering of human interfaces," in psychology, that of "application of theories of cognitive processes and the empirical analysis of user behavior," in sociology and anthropology, that of "interactions between technology, work, and organization," and in industrial design, that of "interactive products."

Thus HCI "studies a human and a machine in communication" and "draws from supporting knowledge on both the machine and the human side" [47]. On the human side, cognitive psychology has always been concerned with the studies of human information processing and performance. For example, Pelz and his group focus on visual perception in everyday life by examining the eye movements of subjects as they perform complex tasks in a natural environment in order to explore the process of how a human stores and recovers information and how a human uses that information in planning and guiding actions [5]. Hewett *et al.* proposes in [47] that "it is important to understand something about human information-processing characteristics, how human action is structured," such as models of cognitive architecture and human action, phenomena and theories of attention, vigilance, motivation, and learning and skill acquisition. He further believes it is important to learn about "the nature of human communication such as aspects of natural languages, specialized languages" and about "anthropometric and physiological characteristics of people and their relationship to workspace and environmental parameters," such as human cognitive and sensory limits, display technologies and design, fatigue and health issues, and design for stressful environments and for the disabled.

On the computer side, some specialized machine components play an important role in interacting with humans. Several topics connected with these components are covered in [47]: (1) input and output devices including virtual devices; (2) dialogue techniques (dialogue input/output, interaction techniques); (3) dialogue genre (style and esthetics, interaction metaphors); (4) computer graphics (2D/3-D geometry, graphics primitives and attributes, solid modeling, and color representation); and (5) dialogue architecture (multi-user/look-and-feel, screen imaging models, window manager models). Myers [73] considers as interactions in HCI those important innovations in computer science such as direct manipulation (visible objects on the screen directly manipulated with a pointing device), mouse pointing device, multiple tiled windows, drawing programs, text editing, spreadsheets, HyperText, CAD, Multi-Media, 3D system, VR (Virtual Reality), AR (Augmented Reality), and natural language and speech.

2.1.2 HCI Modalities

The keyboard and mouse have traditionally been the basic and most common devices in information exchanges between human and machine in computer systems. With the development of HCI and its growth, new modalities for HCI have received great attention and are being developed. HCI is now trying to take as its modalities the natural means that humans employ to communicate with each other: vision, hearing, touch, smell, and taste. As Sharma *et al.* pointed out [83], "Almost any natural communication among humans involves *multiple, concurrent modes of communication*," therefore "people prefer to interact multimodally with computers." They considered four basic questions about multiple-modalities for HCI: (1) *why* integrate multiple modalities; (2) *which* modalities to integrate; (3) *when* to integrate multiple modalities; and (4) *how* to integrate multiple modalities. Here we introduce only the first two topics. As an answer to the first question, they proposed three reasons: (a) Practical reasons: the mouse-keyboard-based systems are unnatural and cumbersome; the current advanced single-modality HCI lacks robustness and accuracy; and multiple modalities (particularly with redundant input) in HCI allow physically or cognitively handicapped people access to computers. (b) Biological reasons: the integration of multiple sensory modalities is the most common thing in the natural world. (c) Mathematical reasons: the optimal ways of integrating different sensory data can produce the best detection rates in the area of target detection. A system using single modality may not reduce the uncertainty for decision making. Furthermore, "it is statistically advantageous to combine multiple observations from the same source because improved estimates are obtained using redundant observations," and "multiple types of sensors may increase the accuracy with which a quantity can be observed."

Sharma *et al.* also discussed the second question under the two categories: *human-action modalities* and *computer-sensing modalities*. In the former case, *hand movements* have been the most exploited modality because of the dexterity of the human hand. The corresponding interface devices are keyboard, mouse, stylus, pen, magnetic wand, joystick and trackball. The use of *hand gestures* is the next human-action modality: from simple pointing to manipulative gestures to more complex symbolic gestures (for example, American Sign Language). In this category are the relevant interface glove-based or video-camera devices. Another dominant human-action modality is the production of sound, especially spoken words, which is deeply connected with one visible action—lip movement. Recently, eye movements, facial expression, and body motion have received more and more attention.

In the case of computer-sensing modalities, five kinds of modalities have been proposed. (1) Position and motion sensing: keyboard for typing input; mouse, light pen, stylus, and tablet for 2D plane input; joystick and trackball for 3D sensing; and many other advanced devices (particularly those in the field of computer vision) for locating and tracking the eye/gaze, head/pose, hand/fingers, and even the body movements. (2) Audio sensing: microphones are used to sense the sound waves, and the techniques of automatic speech recognition can interpret speech, the most natural human-action modality for HCI. It is well known that the visual sensing modality of lip motion can improve the recognition rate for speech [91, 92]. (3) Tactile and force sensing: force sensing by using appropriate haptic devices plays an important role in "building a proper feel of *realism* in virtual reality." (4) Neural sensing: monitoring of brain EEG activities can give one more computer-sensing modality. Also, brain electrical impulses can be used for brain-activated control (BAC). It is particularly promising in the field of aircraft piloting and for the disabled. (5) Visual sensing: so many and various human-action modalities can be incorporated into HCI just by using a video together with a set of techniques. These modalities include the very areas of computer vision: hand gestures, lip movement, eye/gaze tracking, facial expressions, head pose, and other body movements. In the following sections we will discuss in detail the current developments of those computer vision topics and how those modalities are integrated into the HCI context.

2.2 Research on Modeling the Human Body

The human body is too complex an organic structure to be described clearly in one section. In the research areas of HCI and computer graphics related to the studies of the human body, the emphasis is on how to apply the biomechanical features of the body to model a virtual human and to simulate the body movements. In this section, the results from medical studies on the human body will be briefly reviewed first. Then body modeling will be discussed in the computer graphics domain. Because the hand plays a crucial role in human gesturing, the biomechanics and modeling of the hand will be presented in detail.

2.2.1 Articulated Structure and Joint Motions

The skeletal structure of the human body is often used to describe the body motion. The bones consist a highly hierarchical system of rigid objects. During motion the following occurs: during motion, (1) the objects don't change shape; (2) every object moves (rotates) around its axes at motion centers in its own local coordinate system; (3) at a joint between two adjacent objects, one object attaches to the other, and the two separate coordinate systems share a relationship through this attachment. Thus the overall movement of the whole body can be represented as a set of separate motions of individual objects whose local coordinate systems are in the hierarchical structure defined by the human body system.

The first step in modeling the human body and simulating of its movements is to describe the body joints mathematically, specifically, the joint locations, motion centers and axes, and the relationships among the joints. The motions of a joint are measured from a *zero starting position*, which varies with different body joints. There are three general types of joint motions [78]:

1. *One plane* (or *degree*) *freedom* of motion, or the hinge joint, has natural motion in one direction only from the zero starting position. An example is the flexion and extension (away from and return to the zero position) of middle finger at the proximal interphalangeal joint.

2. *Two plane* (or *degree*) *freedom* of motion is motion about the joints with natural motion in two planes originating from the zero starting position. Wrist movement (flexion and abduction) is an example.

3. *Ball and socket* joint motion consists of three-dimensional, compound or rotatory motion. A shoulder is a typical example of this type of motion. Strictly speaking, there is no such motion in the hand, but we can say that thumb motion at the carpometacarpal joint is a three-dimensional motion.

In this subsection, brief descriptions will be given of the motions of the following joints: elbow, forearm, shoulder, and cervical spine. Hand, thumb and finger motions are explained in the next subsection. Detailed descriptions and specifications of these and other human joints can be found in [78].

- **The elbow**
  - *Zero starting position*: The extended straight arm.
  - *Motions*: Natural motion is present in flexion. The opposite motion to flexion is extension. An unnatural one (beyond the zero starting position is hyperextension.

- **The forearm**
  - *Zero starting position*: The vertical upright position (or "thumbs up") with the forearms at the side of the body and the elbow flexed 90 degrees.
  - *Motions*: Pronation and Supination (0 to 90 degrees).
- **The wrist**
  - *Zero starting position*: The extended wrist in line with the forearm.
  - *Motions*: The natural motions are flexion/extension, radial/ulnar deviation, and some degree of rotatory circumduction.
- **The shoulder**
  - The arm at the shoulder has an almost complete range of global motion (360 degrees). The motions (forward flexion, backward extension, abduction) are often described within the coronal plane, sagittal plane, vertical plane, and horizontal plane.
  - *Motions*: (1) Vertical or Upward Motion: abduction/adduction and forward flexion / backward extension. (2) Horizontal Motion: flexion and extension. (3) Rotation with arm at side of body. (4) Rotation in abduction. (5) Glenohumeral motion. (6) Motions of the shoulder girdle: flexion/extension and elevation.
- **The cervical spine**
  - *Zero starting position*: The correct standing or sitting position.
  - *Motions*: (1) Flexion and extension. (2) Lateral bend. (3) Rotation.

2.2.2 Methodologies of the Modeling

In human body modeling and animation simulation, there are two basic approaches: (1) record the motion by using motion capture systems and then modify or re-target such a motion to create individuality, and (2) create computational models that are controlled by a few parameters. Researchers [46, 90] have shown that the first method is tedious and difficult to implement and that the computational models, which give the body posture at specific times and thus are suited for physically-correct dynamic simulation, can offer promising a solution to modeling and animation of a virtual human. Badler *et al.* [30, 31] studied consistent parameterizations for gesture and facial actions and described a parameterized action representation that allows a virtual agent to act, plan, and reason about its actions or the actions of others.

Modeling the human body from the view of computer graphics can be classified into four categories [76]:

- *Stick figure models* consisting of a hierarchical set of rigid segments (limbs) connected at joints. The complexity of these models depend on the number of limbs and joints involved.

- *Volume models*, using a collection of elementary volume primitives (such as ellipsoids, spheres, cylinders) to approximate the structure and the shape of the body. Although better results can be obtained, these models lack adequate control mechanisms for a large number of primitives during animation.

- *Surface models*, containing a skeleton and an external shape (the skin). The problem with this kind of model is the difficulty in controlling the evolution of the surface across joints.

- *Multi-layered models*, containing the skeleton layer, intermediate layers (muscles, fat, bones), and the skin layer to simulate the body animation consistent with human physical aspects. This modeling is difficult, but currently it is considered necessary for a realistic human body animation.

The last two types of modeling are closely related to the problem of deformation modeling which is the most difficult and challenging topic in the creation of lifelike virtual human figures. We can say the key challenge in human body modeling is how to solve the deformation around the body joints and on the body surface when the body parts move.

Multi-layered modeling with deformation

There are three basic steps in multilevel deformation modeling [76]: creating a rigid body for the skeleton, modeling muscle design and deformation, and generating skin. The first step considers the basic structure modeling: the definition of the joints, their positions and orientations, and the geometric model that describes the body hierarchy.

Next, muscles are attached to bones across joints and work like springs along *action lines* which have a fixed endpoint and a movable endpoint. All the muscle (surface) points, producing the mesh for the deformable model, are controlled by three different forces: elasticity force, curvature force, and constraint force. The application of those forces gives new positions for the mesh model.

The third layer is equivalent to the human skin. To smooth the skin surface, B-spline patches are used which are sampled on semi-regular cylindrical grids with a ray-casting method.

A semi-automatic technique for creating animals with the consideration of their anatomical structures was proposed in [84] and [98]. Basic structure (ellipsoids) is used for the creation of underlying body parts such as bones and soft tissues, and polygons are used for the skin, which is automatically generated from the underlying components follows their movements [98]. As in [84], a sparse set of feature points about an animal is used as input data for a canonical model for deformation. First, the input data produce the joint hierarchy and associated body components. Next, a denser set of feature points is generated from these components. Finally, the skin representation can be deformed from the new feature points with a segmented interpolation approach. Although these two methods were used for the animation of animals, they can be applied to human body modeling.

An approach to human figure modeling (particularly musculature) is presented in [81]. The influence of musculature on surface form is considered, and muscle models are developed to automatically react to the changes in the posture of an underlying articulated skeleton. The anatomy-based models were applied to the torso for simulating the behavior of skeletal muscles in humans and to the arm and can be applied to create other surface forms such as bones and fatty tissues.

Surface modeling with deformation

Complex geometric shapes can be generated from current modeling software and a variety of 3D scanning data. The problem is how to modify (automatically) the shapes once they have been created. One paper provides a methodology for efficient run-time interpolation between multiple forms to leverage artist-generated source material [86]. With a set of example forms (surface vertices's), a shape (a continuous range of forms) will be generated, thus producing smoothly skinned figures. This method can be used in interactive applications such as games. Another skinning method for producing accurate approximations of deforming characters is proposed in [71]. It starts with an arbitrarily rigged character in an animation system as an input, that is, a set of examples of skeleton configurations paired with the deformed geometry as static meshes. These sample data are used to fit the parameters of a deformation model that can quickly build character skins.

A method combining skeleton-driven deformation and shape interpolation is introduced in [58]. First, a *pose space* is defined by an underlying skeleton or a more abstract system of parameters; next, a uniform representation of deformation is identified as mappings from a pose space to displacements in the object's local coordinate systems; finally, a single unified approach can be used to accomplish previously disparate deformation types. Another method that considers both skin geometry and skeleton is called *multi-weight enveloping* [95]. A set of training data (coming from existing handcrafted skin animation) is used to calculate coefficients (weights) of a deformation equation by using a modified least-square's fitting technique. The equation can then generalize the skin movement for sequences of animation processes. This multi-weight enveloping offers a concise representation for skin movement and provides a practical way for animation, which can fit well into film production.

Human body scanned data was used as input for calculation of articulated body deformation in [28]. The example data consists of range scans of a human body in several different poses with the use of markers, from which a kinematic skeleton can be constructed and the pose of each scan can be identified. Then, a mutually consistent parameterization of all the scans is created with a possible subdivision surface template. The displacements from this surface are used to represent deformations. The combination of the range scans can be obtained by applying k-nearest-neighbor interpolation in the pose space. This method has been used for the animation of the movement of a human upper body [28].

Other methodologies of deformation

An interactive simulation of deformable complex characters has been done with a combination of controllable underlying skeleton and coarse volumetric control lattice that aligns with the bones of a simple skeleton [38]. The control lattice is used to provide the structure for the calculation of deformation with the finite element method. The computation is accelerated by associating regions of the volumetric mesh with particular bones and then performing locally linearized simulations. A hierarchical basis on the control lattice is defined so as to adapt the level of detail. Another interactive geometric deformation technique is proposed in [85] with the application of space curves and implicit functions that aggregate to deform an object. *Wire* curves define an object and shape its deformable features, while *domain* curves define the domain of deformation about the object. A wire with a set of domain curves thus provides a new basis for an implicit modeling primitive. Wires give a coarse geometric representation of an object; the combination of wires and domain curves provides a new way to outline the shape of an implicit volume in space. This method has been implemented as a module in *Maya* software [15].

Vertex deformation is a common technique in simulation of joint movements of the human body. The surface vertices of an object "move" interactively in response to some control mechanism at the joints. A medial-based vertex deformation is proposed in [35]: the vertex weights associated with deformation can be computed by employing convolution to the medial axis and surface of an object. It was applied to the modeling of upper joints of the human body and achieved good results.

A practical rendering technique was applied in real-time simulation of dynamic deformations of character animation in [51]. It is called dynamic response texture (DyRT) mapping. DyRT has to use precomputed modal vibration models stored in graphics hardware memory and driven by inputs from rigid body motion. It can map to any conventional animation as an optional rendering stage and produces real time renderings with negligible main CPU costs.

2.2.3 The Human Hand

Humans have five senses: sight, hearing, smell, taste, and touch. With the exception of the sense of touch, animals enjoy far more acute sensory faculties. "We ought to define the hand as belonging exclusively to man," Sir Charles Bell acknowledged in 1885 [34, p. 26] after having thoroughly studied and painstakingly compared the animal extremities that have thumb and fingers with the human hand, "corresponding in sensibility and motion with that ingenuity which converts the being who is the weakest in natural defense, to the ruler over animate and inanimate nature." The hand's capability of executing "whatever man's ingenuity suggests" corresponds with human's superior mental intelligence [34, p. 157]. A "lovely hand is the product of a lovely mind. The involvement of the hand can be seen in the face which is in itself a sort of mirror to the mind" [74, p. 21].

Naturally, humans have made full use of the hand—"the ready instrument of the mind" [34]—in not only administering the harsh natural environment but also enriching human spiritual life. For example, ancient men did have meditative thought on the hand (Figure 2.1a). For the past few decades, computer systems have used a peripheral devices—a mouse—to get man's message communicated with the hand (Figure 2.1b). And in the future, very human hand gestures will directly feed information in human-computer interaction (Figure 2.1c).

(a) (b) (c)

Figure 2.1: The human hand as intermediate tool: (a) A red hand silhouette painted on a stone slab, about 10,000-8,000 years ago (from [1]); (b) Mouse/hand as a medium between computer and human (from [9]); (c) Hand gestures provide input in HCI (from [2]).

This "ready instrument of the mind" deserves a thoughtful look at its functionality and gesticulation.

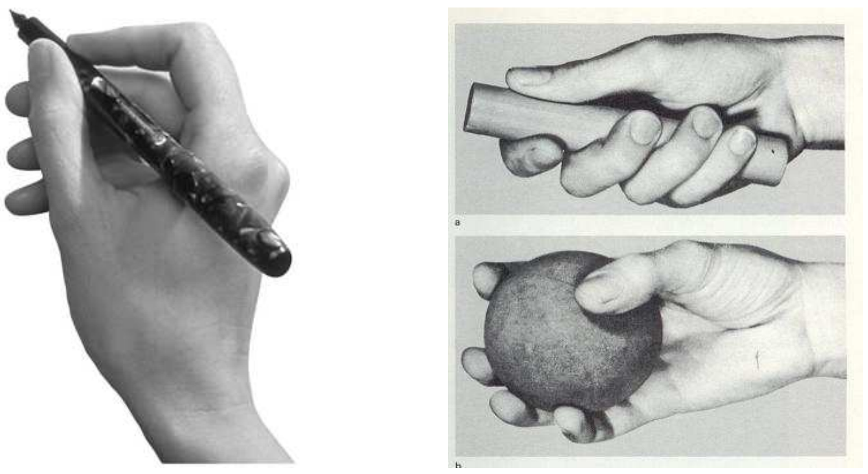
Function of the Hand

"The hand at rest is beautiful in its tranquility, but it is infinitely more appealing in the flow of action" [74, p. 21]. The graceful movement of the thumb and fingers demonstrates the practical functionality of hand, which can be roughly seen through opposition, prehensile and non-prehensile movements.

Opposition is the most important movement of the human hand. The thumb plays a significant role in opposition. Napier gave a definition for it: "Opposition is a movement by which the pulp surface of the thumb is placed squarely in contact with – or diametrically opposite to – the terminal pads of one or all of the remaining digits" [74, p. 68]. An example of opposition can be seen in Figure 2.2a.

Generally, there are two classes of hand movement of which a human is capable: prehensile and non-prehensile. Some of the non-prehensile movements include pushing, lifting, and tapping movements of the whole hand.

Prehensile movements are typified by gripping or pinching between the digits and palm an object, whether fixed or free. Napier conjectured that there are two main prehensile patterns (power grip and precision grip, Figure 2.2b) and two subsidiary prehensile patterns (hook grip and scissor grip). It should be noted that the type of grip used in any activity is a function of the activity itself and does not depend on the shape or size of the object gripped [74, p. 77].



(a) (b)

Figure 2.2: Functions of the hand: (a) Opposition in practice (from [10]); (b) Power grip (upper) and precision grip (lower) (from [74, p. 75]).

Gestures of the Hand

David McNeil did a careful study of people's spontaneous gestures that accompany their speaking and pointed out that "gestures are an integral part of language as much as are words, phrases, and sentences—gesture and language are one system" [66, p. 2].

Napier studied the evolution, mechanics, and functions of the human hand and came to a conclusion that "if language was given to men to conceal their thoughts, then gesture's purpose was to disclose them" [74]. McNeil went on to explore how human thoughts are disclosed in gestures and classified human gestures into four types of gesture: *iconics, metaphorics, beats, deictics*, and *cohesives*, which is a combination of any of the other four gestures [66].

A gesture can be analyzed in its time and space patterns. A gesture's cycle begins with gesture preparation in which the hand moves to a rest position, culminates in a stroke period during which the full meaning of the gesture is expressed, and ends with retraction in which the hand returns to a rest position. Different types of gestures tend to congregate in different regions in the gesture space which can be visualized as a shallow disk in front of the speaker [66].

Whole-hand Input Issues for HCI

Sturman defined whole-hand input as "the full and direct use of the hand's capabilities for the control of computer-mediated tasks" [88]. Whole-hand input refers to the direct measurement of hand motion (finger flexions/abductions/rotations and/or whole hand rotation/displacement) rather than measurement of the motion of a device manipulated by the hand. Sturman also distinguished three important issues on whole-hand input: *appropriate use, control design*, and *device.*

The effectiveness of whole-hand action can be evaluated with the experiments on low-level physiological capability, task performance and the cognitive basis of hand functions. Fitts' law [63, 88] and "steering law" [11, 26] can be used for this purpose.

2.2.4 Biomechanics of the Human Hand

This section is an overview of results from medical studies of the hand: anatomy, biomechanics, and constraints of the human hand.

Anatomy of the human hand

The human hand can been seen as an organic structure composed of muscles, tendons, and bones. Tendon is the tough cord or band of dense white fibrous connective tissue that unites a muscle with a bone and transmits the force that the muscle exerts, allowing movement at a joint. In the case of the hand, most of the muscle mass lies in the forearm with long tendons which transmit force to the fingers and allow the hand to be light and flexible without sacrificing strength. Hand movement is thus described by the interconnections and interactions of muscles and tendons. Our interest, however, is in the hand skeleton structures. Therefore, we consider hand movement as determined by the position/orientation of the hand bones. A hand skeleton is shown in Figure 2.3.

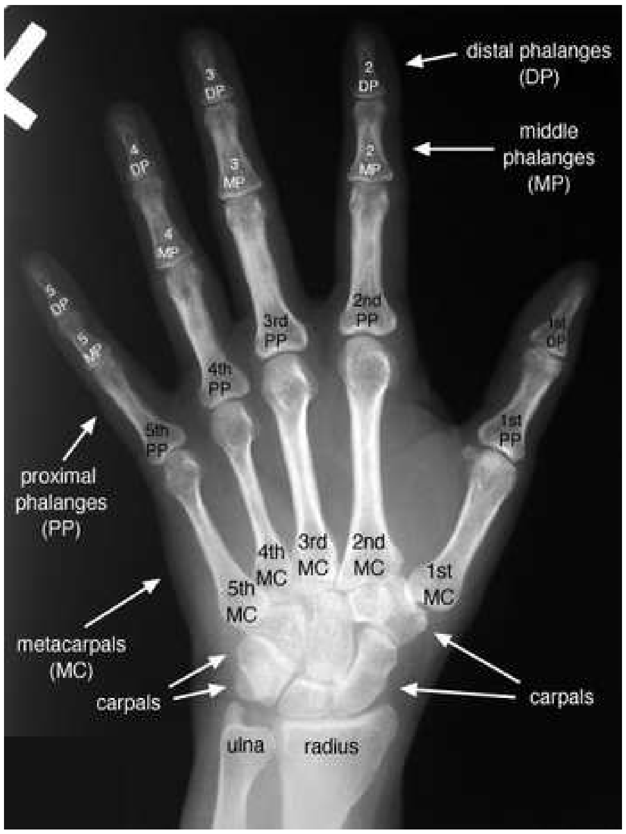


Figure 2.3: Hand bone structure (from [8]).

The hand as a machine

Paul Brand [37] asserted that the hand is more than a machine and that studying the hand with only mechanical principles is to “debase a thing of exquisite beauty and sensitivity to compare it with the crude and smelly things that we call machines.” There are reasons, however, why we have to think of the hand as a machine in order to understand the mechanical principles that can be applied and to be able to produce modified mechanics of hand joints and tendon placements [37]. Brand considered three major aspects of the hand machine: the motor (muscles), the transmission (tendons, bones, joints), and the application (through skin and pulp tissues). The relationship among them is shown in Figure 2.4.

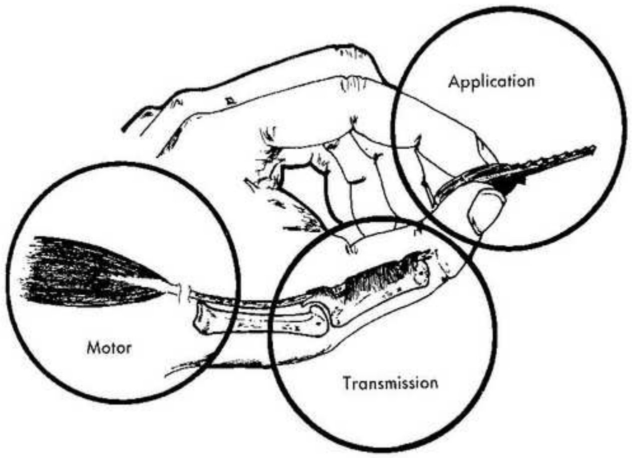


Figure 2.4: The three major elements of hand motion (from [37, p. 9]): muscles serve as motor, tendons/bones/joints used as the transmission, and skin and pulp tissues for application. They are controlled and moderated by motor nerves and by sensation.

The hand joints

Movement at a joint is produced by the contraction of muscles. Hand motion involving different joints requires muscle contraction, tendon transmission, and skin tissue application. Figure 2.5 demonstrates some common motions of hand joints.

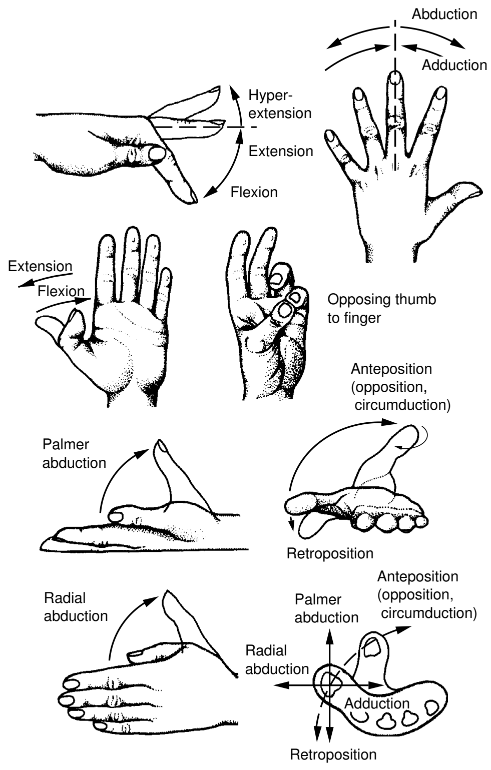
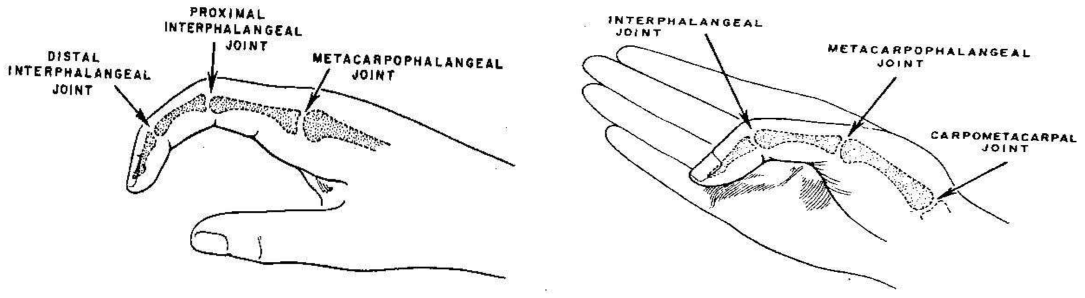


Figure 2.5: Common hand joint motions (from [88] and [78]).

In order to describe and measure specific hand motion, it is important to understand hand joints. Figure 2.6 is the structural diagram for fingers and the thumb from which we can easily identify the joints:


PROXIMAL INTERPHALANGEAL JOINT
DISTAL INTERPHALANGEAL JOINT
METACARPOPHALANGEAL JOINT
INTERPHALANGEAL JOINT
METACARPOPHALANGEAL JOINT
CARPOMETACARPAL JOINT

(a) (b)

Figure 2.6: Finger and thumb joint definition (from [78]).

- Finger joints

DIP	Distal interphalangeal joint
PIP	Proximal interphalangeal joint
MCP	Metacarpophalangeal joint

- Thumb joints

IP	Thumb interphalangeal joint
MCP	Thumb metacarpophalangeal joint
CMC	Thumb carpometacarpal joint

Boimechanics of the hand

From a biomechanical viewpoint, the human hand can be considered as a linkage system of connected bony segments, in which the joints between each phalanx are spanned by ligaments, tendons, and muscles. To study hand motion and its force analysis under various functional activities, a three-dimensional normative model of the human hand was established in [39], based on the anatomical study of normal hand specimens.

Experiments were carried out for four of the finger motions: flexion-extension at distal interphalangeal (DIP) joint, proximal interphalangeal (PIP) joint, and metacarpophanlangeal (MCP) joint, as well as abduction-adduction of the MCP. Six Cartesian coordinate systems were established for a finger to define the locations and orientations of tendons and to describe the joint configurations. At a joint, a pair of coordinate systems are assigned (See Figure 2.7): the proximal system (X_P, Y_P, Z_P) which is located at the approximate center of rotation of the phalangeal and metacarpal head, and the distal system (X_D, Y_D, Z_D), which is a translation of the proximal system to the center of the concave articular surface.

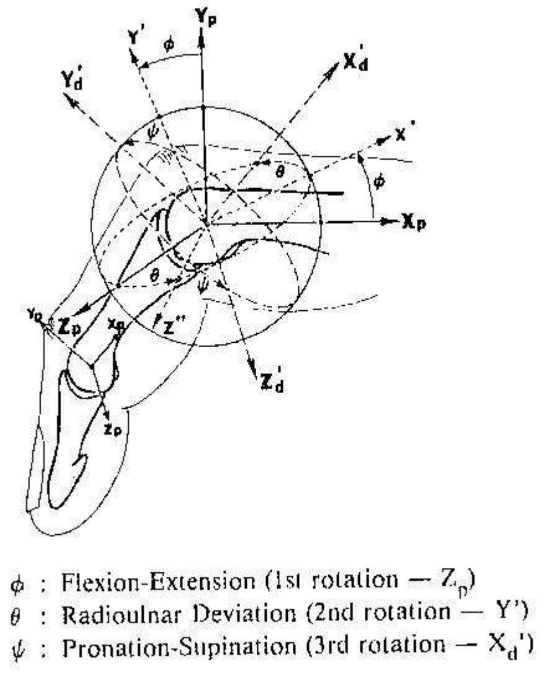

Y_p
Y'
ϕ
X_d'
Y_d'
X'
ψ
θ
ϕ
X_p
θ
ψ
Z''
Z_p
Z_d'
Z_p
ϕ : Flexion-Extension (1st rotation — Z_p)
θ : Radioulnar Deviation (2nd rotation — Y')
ψ : Pronation-Supination (3rd rotation — X_d')

Figure 2.7: Eulerian angle definition for a finger (from [39]).

When the finger is in a functional configuration other than the neutral position, there exists a transformation (rotation and translation) at a particular joint between the proximal coordinate system and the distal system:

$$\begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix} = \begin{bmatrix} c\theta c\phi & c\theta s\phi & -s\theta \\ -c\psi s\phi + s\psi s\phi c\theta & c\psi c\phi + s\psi s\theta s\phi & s\psi c\theta \\ s\phi s\psi + c\psi c\phi s\theta & -s\psi c\phi + c\psi s\theta s\phi & c\psi c\theta \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix} + \begin{bmatrix} X_O \\ Y_O \\ Z_O \end{bmatrix}$$

in which

- (X_D, Y_D, Z_D) = coordinates of a tendon point or components of a vector measured with respect to the *distal* system,
- (X_P, Y_P, Z_P) = coordinates of a tendon point or components of a vector measured with respect to the *proximal* system,

- (X_O, Y_O, Z_O) = coordinates of the origin of the *proximal system expressed in the distal system*,
- θ, ϕ, ψ = Eulerian angles for flexion-extension, radio-ulnar deviation (abduction-adduction), and axial rotation, respectively, (see Figure 2.7), and
- s, c = sine and cosine functions.

Tendon excursion and moment arm at a joint can provide useful information for the consideration of deformation at a joint in hand modeling. The excursion and moment arm are calculated, depending on the pattern of tendon movement. One of the typical patterns happens when a tendon runs in a tendon sheath which holds it firmly in a constant position against the shaft of the bone but allows the tendon to curve smoothly, conforming to the joint architecture (Figure 2.8).

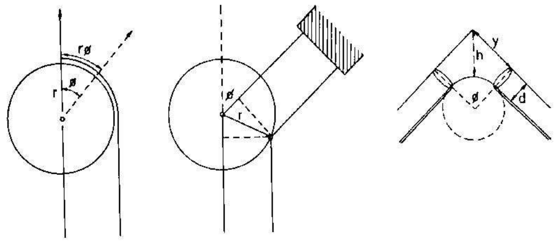


Figure 2.8: Tendon bridging a joint.

Hand joint orientation and range of motion

Forty normal subjects, with no clinical or radiologic evidence of joint disease, were studied with regard to normal finger joint motion (twenty-seven for the thumb case) [39]. The results of the study of normal finger joint motion are summarized in Table 2.1. Ranges for the thumb are given in Table 2.2. All the results in Table 2.1 are mean values.

Table 2.1: Finger joint motion ranges (from [39])

Finger Joint Motion Ranges (degrees)					
Joint	*Motion*	*Index*	*Middle*	*Ring*	*Pinky*
DIP	Flexion	73	80	75	78
	Extension	11.45			
PIP	Flexion	101	103	105	103
	Extension	10-12			6.5
MCP	Flexion	83	90	88	90
	Extension	-22	-22	-23	-34
Total flexion		256	274	268	272

Table 2.2: Thumb joint motion ranges (from [39])

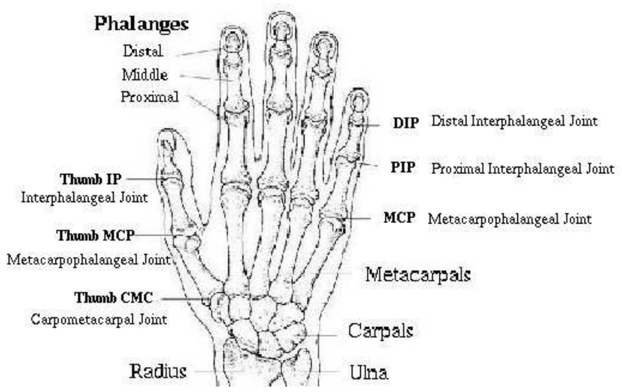
Thumb Joint Motion Ranges (degrees)		
Joint	*Motion*	*Thumb*
IP	Flexion-extension	100 ± 9
	Abduction-adduction	7.5 ± 10
	Rotation	8.4 ± 9
MCP	Flexion-extension	45 ± 16
	Abduction-adduction	8.7 ± 3.2
	Rotation	12.1 ± 4
CMC	Flexion-extension	53
	Abduction-adduction	42
	Rotation	17

The researchers noticed that MCP motion had considerable variability. In flexion, there was a range of $65^o - 107^o$ from neutral position when measured with respect to the metacarpals. The PIP joint motion is more consistent with a smaller range of variation of $92^o - 125^o$. The variation range for the DIP extension angle is $10^o - 23^o$. Abduction-adduction and rotational motions were generally very small for the DIP and PIP joints but significant for MCP joints (not given quantitatively). Variations for the thumb are given in the Table 2.2.

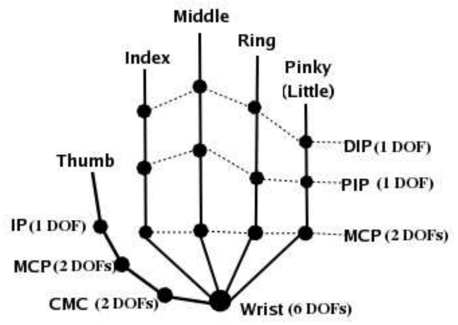
The hand constraints

The human hand, consisting of many connected parts, as the above sections indicate, is highly articulated. However, it is also highly constrained because the hand cannot make any arbitrary gestures. Hand movements can be divided into two categories: natural movements in the hand without external interaction and unnatural movements under external force. The first one is also called active movement and the other, passive movement. Natural movement may produce constrained movement, for example, while you bend middle finger your ring finger automatically bends.

We employ Figure 2.9 to introduce the hand constraints. Each finger has three joints: DIP, PIP, MCP; and the thumb has also three: IP, MCP, CMC. We use T, I, M, R, P to represent thumb, index, middle, ring, and pinky (little). Also, “A-A” stands for “abduction-adduction,” “F-E” for “flexion-extension,” “F” for “flexion,” and θ for degrees of flexion, abduction, *etc.* We can calculate the degrees of free-



(a)



(b)

Figure 2.9: Hand joint representation: (a) The hand joints—They can be seen as a structure of 17 active joints with 23 degrees of freedom (adapted from [74, p. 29]); (b) The stick representation of the hand joints.

dom (DOFs) in the hand following a robotics conception. There are 27 DOFs: 1

DOF for DIP/PIP/IP (flexion-extension), 2 DOFs for MCP/CMC (flexion-extension and abduction-adduction), 6 DOFs for the wrist (rotation or supination-pronation, bending or flexion-extension, side-side or radial-ulnar deviation, and 3 DOFs for displacement). Thus, each finger has 4 DOFs, the thumb has 5 DOFs, and the wrist has 6 DOFs.

Furthermore, since finger flexion might cause flexion of the adjacent fingers and a finger's extension is hindered by the flexion of others, two phrases, *static joint angle limit* and *dynamic joint angle limit*, were introduced in [57]. These terms distinguish, respectively, between the joint angle limit calculated for all possible hand configurations and that calculated with the other joints in some specific configurations.

Three types of hand constraints are summarized in [61]: (1) the constraints limited by the finger motion as a result of hand anatomy, or static constraints, *i.e.*, joint motion ranges, (2) the constraints imposed on joints during motion, or dynamic constraints, for example, the intra-finger and inter-finger constraint, and (3) the constraints applied in performing natural motion, for example, curling all the fingers at the same time to make a fist.

Following is an enumeration of the hand constraints from studies on finger and thumb motion:

1. The four fingers are planar manipulators with the exception of the MCP joints [57].
2. The DIP and PIP joint flexions have a dependency [80] represented by:
$$\theta^F_{DIP} = \tfrac{2}{3}\theta^F_{PIP}$$
3. The MCP joint in the middle finger displays little abduction/adduction [57, 80]:
$$\theta^{A-A}_{MCP(M)} = 0$$

4. The joint angle limits of the MCP joints depend on those of the neighboring fingers according to the following [57]:

    - $dmax(\theta^F_{MCP(I)}) = min(\theta^F_{MCP(M)} + 25, smax(\theta^F_{MCP(I)}))$
    - $dmin(\theta^F_{MCP(I)}) = max(\theta^F_{MCP(M)} - 54, smin(\theta^F_{MCP(I)}))$
    - $dmax(\theta^F_{MCP(M)}) = min(\theta^F_{MCP(I)} + 54, \theta^F_{MCP(R)} + 20, smax(\theta^F_{MCP(M)}))$
    - $dmin(\theta^F_{MCP(M)}) = min(\theta^F_{MCP(I)} - 25, \theta^F_{MCP(R)} - 45, smin(\theta^F_{MCP(M)}))$
    - $dmax(\theta^F_{MCP(R)}) = min(\theta^F_{MCP(M)} + 45, \theta^F_{MCP(L)} + 48, smax(\theta^F_{MCP(R)}))$
    - $dmin(\theta^F_{MCP(R)}) = min(\theta^F_{MCP(M)} - 20, \theta^F_{MCP(L)} - 44, smin(\theta^F_{MCP(R)}))$
    - $dmax(\theta^F_{MCP(L)}) = min(\theta^F_{MCP(R)} + 44, smax(\theta^F_{MCP(L)}))$
    - $dmin(\theta^F_{MCP(L)}) = max(\theta^F_{MCP(R)} - 48, smin(\theta^F_{MCP(L)}))$

    where *dmax, dmin, smax*, and *smin* are the variables for the dynamic and static joint angle limits with *min* and *max* being the minimum and maximum values.

5. There exists [57]:

    $$dmax(\theta^{A-A}_{MCP}) = k \times smax(\theta^F_{MCP})$$

    where $k = \left(1 - \frac{1}{smax(\theta^F_{MCP})}\right)\theta^F_{MCP}$

6. The thumb has the following equations [80]:

    $$\theta^F_{CMC} = 2(\theta^F_{MCP(T)} - \tfrac{1}{6}\pi)$$
    $$\theta^{A-A}_{CMC} = \tfrac{7}{5}\theta^{A-A}_{MCP(T)}$$

7. The MCP and PIP joints of a finger have a dependency [56] represented by:

    $$\theta^F_{MCP} = k\theta^F_{PIP};\ 0 \le k \le \tfrac{1}{2}$$

    where $k = \frac{1}{2}$ is suitable for dynamic cases.

8. There exist "finger planes" [40] for each finger: the five points—wrist joint, MCP joint, PIP joint, PIP joint and the finger tip—are coplanar (there may be a different wrist point for each finger).

9. There exists a "weak constraint" on the thumb [40]:

$$\theta^{F}_{IP(T)} = a\theta^{A-A}_{MCP(T)};\ a \geq 0$$

10. There exists a "thumb plane" [40]. Four points—CMC joint, MCP joint, IP joint, and thumb tip—are coplanar.

11. There exists a "palm plane" [40]. All the four MCP joints of the four fingers are located in a plane. It is assumed that this plane is perpendicular to the "finger plane" of the middle finger.

12. The axis of the thumb and the axes of the two phalanges (distal and middle) of each of the fingers converge to a point when the hand is clenched into a fist [57] (Figure 2.10). The process of the convergence means that the abduction and adduction angles continuously decrease as the flexion angles of the MCP joints increase.

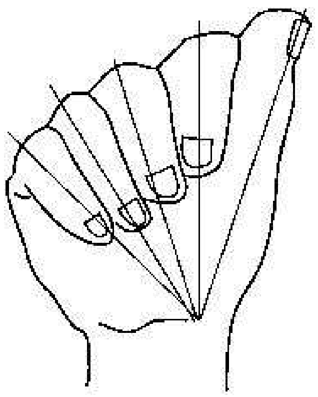


Figure 2.10: Convergence of fingers when the fist is clenched (from [57]).

One advantage of applying hand constraints is computational efficiency. Hand constraints reduce the size or dimensions of the search space and thus making the estimation of hand postures more effective. Incorporating constraints in hand modeling allows for synthesizing natural hand motion and thus producing realistic hand animation in hand animation and hand sign language analysis. Instead of considering the total 27 DOFs, Chua *et al.* [40] simplified the hand model by reducing it to 12 DOFs and Lin *et al.* [61] reduced it to 15 DOFs. There is a trade-off, however, between the simplification and modeling of the hand and the simulation of the hand movements. The reduction in DOFs cannot accurately describe the true movements of the fingers and thumb in real life and will make the gesture simulation less realistic.

2.2.5 Modeling the Hand

The human hand, consisting of bones, ligaments, muscles, tendons, soft tissues and skin, is an organic component of the human body. It can be seen as a highly articulate, exceedingly complex mechanical structure. Bones are linked at joints while muscles and tendons and ligaments provide the architectural basis for the flexibility of the hand. Hand modeling is a challenging topic belonging to the area of modeling, deformation, and animation of the human body, and human modeling and animation bring together state-of-the-art technologies [29, 30, 31, 32, 36, 70, 90].

Roughly speaking, hand shape models can be categorized into three groups [99]: geometric models, physical models, and statistical models. The main issue in the latter two models is the consideration of the deformation of hand: a physical hand shape model focuses on an explicit representation of hand deformation under the internal and external forces applied to the hand, while a statistical hand model implicitly learns the deformation of the hand through a set of training examples (*i.e.*, 2-D images of hand gestures). As for the geometric hand shape models, a wide variety of choices are available, depending on the complexity, from simpler parameterized geometric shapes (such as a cylinder of super-quadric), to 3-D polygon meshes, to complicated spline-based geometric surfaces.

An exploration has been done into capturing the intricate nature of the fingers during natural hand movement [42]. The hand model is an interactive module that improves the realism of arbitrary hand animations. A data-driven algorithm is implemented to add sympathetic finger motion to arbitrarily animated hands. A procedural algorithm is then used to generate the motion of the fretting hand playing a given musical passage on a guitar. In the model, the joint velocity and wrist and arm postures are considered to allow better handling of dexterity constraints and provide better motion dynamics as the fingers move from note to note.

A human hand model is presented with underlying anatomical structure [27]. The construction and animation of the model is based on a reference hand model, and its animation is controlled by muscle contraction values. A physically-based hybrid muscle model is used to convert these contraction values into movement of skin and bones. Pseudo muscles control the rotation of bones (based on anatomical data and mechanical laws), and geometric muscles deform the skin tissue (using a mass-spring system). The hand animations are automatically obtained through the finger movements and skin deformations. In the creation of individual hand models from photographs, a radial basis warping function is built from the correspondence of feature points and applied to the structure of the reference hand model, thus the hand model is instantly deformed.

A method called *EigenSkin* is introduced for real-time large-deformation character skinning and obtained good results in hand modeling and animation [54]. Principal components of the deformation influences are precomputed for individual kinematic joints (instead of storing large amount of displacements for key hand poses). Error-optimal eigenbases are thus constructed for describing each joint's deformation subspace. The pose-dependent deformations are represented in terms of these reduced eigenbases and the precomputed coefficients of the eigenbasis are to be interpolated at runtime. A finite element model of the human hand is created to illustrate this *EigenSkin* method and demonstrate subtle nonlinear skin deformations.

2.2.6 Modeling the Upper Body

A major work in virtual human modeling is the representation and simulation of of the human upper body movements. Investigations have been done toward biomechanical modeling and simulation of the human upper limbs by Walter Maurel [64]. Following are brief descriptions of his work unless otherwise referenced.

A 3D biomechanical, musculoskeletal human upper limb model is designed to simulate the dynamic musculoskeletal animation of the human upper limb with the help of a finite element simulation software of soft tissue deformation and muscular contraction. Then the anatomical and biomechanical modeling of the scapulo-thoracic constraint and the shoulder joint sinus cones is embedded into the limb model. With the implementation of inverse kinematics, realistic animation is obtained of a virtual skeleton and an anatomic musculoskeletal body model.

Forward dynamics can be used for the simulation of the upper limb motion. Its input is from the muscle forces exerted on the skeleton, which are calculated from the forces and torques at the joints (when in motion) with inverse dynamics. Because of the complexity of the musculature, the muscle force contributions cannot be definitely determined. Some static or dynamic analyses of muscle topology are needed to predict their values.

The joint boundaries of the upper body are modeled like conic shapes with elliptic bases. To some joints, optional adjunction of cone data structure should be created for the implementation of specific functions for bounding the skeletal segments within their respective conic mobility domains. Particular shapes of the elliptic bases of the cones need interactive adjustment of the cone shapes and testing of the associated method for bounding the limb motion within the cone.

In the virtual human modeling, the shoulder, one of the most complex articulations of the human body, is usually modeled as a simple 3 DOF rotational joint, which often leads to unrealistic deformation. An extended shoulder model was proposed in [65] which includes the modeling of the scapulo-thoracic constraints as a scapular end-effector maintained on a thoracic ellipsoid using inverse kinematics and the modeling of the joint sinus cones for providing realistic boundaries to the shoulder joints.

Following is a partial list of the many other methodologies used for modeling the upper body and joints: a medial-based vertex deformation [35], an example-based deformation from range scan data [28], least-squares approximation (multi-weight enveloping) technique for skin animation [95], an accurate character skin building from example inputs of static deformed meshes and their skeletal configurations, and skinning from the generation of range flows using stereo and temporal matching between successive images [75].

2.2.7 Modeling the Head

Human head modeling and facial animation are two of most challenging topics in computer graphics. According to [6], facial modeling and animation includes the following issues: (1) Data acquisition—how to obtain the facial geometry of real individuals. (2) Muscle models—accurate modeling and simulation of facial expressions, which are formed by orchestrated contraction of the underlying muscles. (3) Simulation of skin—how to mimic the important characteristics of facial skin (such as bulging, expressive wrinkles and furrows), which is composed of several layers of skin tissues (each of these having specific biomechanical properties). (4) Textures and rendering—how to use rather coarse triangle meshes for fast real-time rendering for the simulation of skin structure and reflection properties. (5) Multiresolution surface—exploiting the multiresolution hierarchy for quick skin/tissue simulation on a coarse base mesh and reconstruction on the deformed mesh. (6) Speech synchronization—generating lip movements that correspond to the recorded speech. (7) Anthropometric modeling—making use of statistical (anthropometric) measurements on populations in different ages, sexes, and ethnicities to create a variety of head models.

A physical facial model is presented in [100] that provides high fidelity for facial animation while optimizing the computation. The model incorporates a physiologically-based approximation of facial skin tissue (including the underlying skull structure) and thus has a multilayer biomechanical structure. This dynamic skin model uses nonlinear springs to simulate the nonlinear visco-elastic behavior of soft tissue. To simulate distribution of the muscle force (due to muscle contraction), different types of muscle models have been used. The presence of the skull constrains the skin movements, thus providing more accurate facial deformation, and guides the interactive placement of facial muscles.

A head model is proposed in [52] that employs anatomical structure and deformation methods to allow adaptation of such a model to individual scan data and even to simulate the human head growth. This versatile construction and deformation method for the head is suitable for real-time physics-based facial animation. With landmark data on skin and skull, the model deforms the head in anthropometrically meaningful ways. The underlying muscle and bone structure is adapted so that the model can use the same muscle contraction parameters.

A system was introduced in [55] that can be used for the design of personality for an emotional virtual human. Five factor models of personality from psychology studies have been adopted in the system: extraversion, agreeableness, conscientiousness, neuroticism, and openness. A Bayesian Belief Network and a layered approach are used for the implementation of the model personality, mood, and emotions.

2.3 Sign Languages

Complexity of language is one of the things that distinguishes the human being from other living beings. We use languages, either in speaking or writing, to express our thoughts and feelings and to convey information. When we speak, we automatically (and most often subconsciously) make gestures including facial expressions. Some studies have suggested that language evolved from manual and facial gestures to signed languages and then to the spoken language with grammar and syntax [41, 87]. Even though this view has not yet been accepted as a general concept, researchers agree that both languages and gestures are representations of a single, deep-seated system in the human being and that they differ only in their modalities of production (oral *vs.* gestural) and perception (aural *vs.* visual) [53, 66, 67].

Sign languages, developed from gestures, are classified as natural languages [59, 94]. Like spoken languages, sign languages have evolved independently of each other and have similar expressive power as that which is inherent in spoken languages, conveying the same subtleties and complex meanings as spoken languages can. They are not the manual versions of oral languages—*i.e.*, American Sigh Language (ASL) is not "English on the hand"— but have intricate compositional structures at all their linguistic levels (*e.g.*, phonology, morphology, syntax, and discourse) in which smaller linguistic units are combined to create linguistic units with higher-level structure (such as sentences constructed from words) [43]. One of the most interesting phenomena involving sign languages is that deaf signers with brain damage suffer from sign-language deficits and that the deficits resemble Wernicke's or Broca's aphasia, associated with spoken language [48].

Although having common abstract characteristics inherent to natural languages, sign, oral, and written languages differ drastically in their outward representations. "Spoken languages are encoded in acoustic-temporal changes—variations in sound over times [48]." Speakers depend on their aural and oral channels for the exchange of information. Written symbols (such as vowel and consonant letters) are used to transcribe the spoken sounds and generate words, and words are arranged in sequential order, following grammar rules to make sentences. Written derivations of spoken languages greatly enhance human life as they allow information to be preserved irrespective of time.

Sign languages, on the other hand, "rely on visual-spatial changes to signal linguistic contrasts [48]." Those that are hard of hearing depend on visual and gestural faculties as their main avenues of communication. There is no standard writing system that can be used to transcribe the signing information, although there exist some notation systems, such as *Stokoe Notation System* [24, 25], *Sutton* [93] and *Laban* [14] *Writing Systems*, and *HamNoSys* [7] *Notation System*. In fact, there are multiple notation systems even in the same sign language [49]. Furthermore, to people with hearing impairments, a writing system of a sign language is like a second language because their native language depends heavily on visual and gestural channels (and cannot easily be mapped into textual symbols).

To illustrate the distinctive features of sign languages, American Sign Language (ASL) will be used as an example in the following sections. Most of the contents are extracted from [94] unless otherwise referenced.

2.3.1 American Sign Language

American Sign Language is a natural language used by about $500,000$—$1,000,000$ people in North America as their primary or secondary language [72]. ASL has developed naturally among a community of users and exhibits all the features of a language: phonology, morphology, syntax, semantics, and so on.

Phonology refers to the study of the smallest contrastive units of language (*e.g.*, the sounds in spoken language). Sign language phonology is the study of how signs are structured and organized. ASL phonological units include five basic parts (parameters): the hand shape, movement, location, orientation of the hand, and the nonmanual signals (NMS), called facial expressions. William C. Stokoe created the first system for describing signs and proposed that signs have three parts (parameters): location, handshape, and movement. He defined a set of symbols for different values of the three parameters (a parameter can have many values: for example, symbols B and C are two values of the parameter *handshape*, meaning a flat hand and curved hand).

According to Stokoe's model, signs consist of three "simultaneously" produced parameters, having a different structure from that of spoken languages. But from Liddell and Johnson's studies on ASL, the signs are composed of hold segments and movement segments that are produced sequentially. Here information about the handshape, orientation, location, and NMS is represented in bundles of articulatory features (similar to the ones in describing sounds in spoken languages). Hold segments (holds) are thus defined as period of time during which all the aspects of the articulation bundle are in a *steady* state, while movement segments (movements) as periods of time during which some aspects of the articulation are in *transition*. We call this model the Movement-Hold Model.

In ASL, the Movement-Hold Model provides the level of detail needed for description of sign structure and of sign process. This model can also illustrate that the fundamental structure of sign language is parallel to the fundamental structure of spoken language and that in ASL there exist phonological processes where sign parts may influence each other and occur in different order: (1) *movement epenthesis* in which a movement segment is added after the last segment of one sign and before the first segment of the following sign, (2) *hold deletion* that deletes holds between movements as signs occur in sequence, (3) *metathesis* in which parts of the segments of a sign change places, and (4) *assimilation* that occurs when a segment takes on the features of another segment before or after it.

Morphology refers to the studies of the smallest meaningful units and how these units consist of new words or signs. There are many different types of morphological processes in ASL to build new words or signs. Typical examples include deriving nouns from verbs, compounding (two signs come together to form a new one), lexicalized fingerspelling (representing the symbols of written English with ASL signs), numerical incorporation (bound morphemes combined to create new meanings, *e.g.*, TWO-WEEKS, THREE-MONTHS, and most importantly, the classifier predicate, which consists of a movement root and a classifier handshape (for example, VEHICLE-DRIVE-BY—a 3 handshape moving from right to left in front of the signer with the palm facing in—is a classifier predicate. You can describe "how a car, boat, or bicycle drove past" by using the sign CAR, BOAT, or BICYCLE followed by the sign VEHICLE-DRIVE-BY).

ASL has its syntax or grammar—a finite set of rules used to produce an infinite set of sentences. Like any spoken languages, its major lexical categories include nouns, predicates, adjectives, and adverbs while the minor lexical ones comprise determiners or pronouns, auxiliary verbs, conjunctions, and prepositions. Of ASL sentences, the basic word order with intransitive verbs is subject-verb; the order with transitive verbs is subject-verb-object. Other word orders are possible, sometimes with the accompaniment of nonmanual signals. Basic ASL sentence types are: (1) questions, (2) negation, (3) commands, (4) topicalization (object of a sentence moved to the front of the sentence), and (5) conditionals.

For convenience, an ASL sign is represented with small capital letters for the English word that corresponds that the ASL sign, a method which is called *glossing*—selecting an appropriate English word for signs in order to write them down. For example, CAT, CUTE are used to represent ASL signs *cat, cute.* Glossing is like translation of English words but contains more ASL features by combining more *markers* (such as plural markers or nonmanual signal markings) with the small capital letters.

2.3.2 Computer-related Sign Language Studies

Using a computer to study sign languages is not a new concept. In the early age of computer graphics, an interactive computer graphic system was introduced to analyze and model the complex hand and arm movements of sign language [62]. Through the reconstruction and manipulation of actual sign movements, this system was designed to convey ASL essential grammatical information with a line drawing as its visualization output.

With the advancement of computer technology, the *Dictionary of American Sign Language on Linguistic Principles* (DASL [24, 25]) has become the *Multimedia Dictionary of American Sign Language* (MM-DASL [96, 97]), presenting ASL signs in full-motion (video of ASL entries), enabling users to search for words in two ways by entering English words or ASL pronunciation criteria.

Live-action video clips combined with implementation of graphical user interfaces significantly assist the ASL studies. For example, *SignStream* is a multimedia database tool designed to facilitate linguistic and computer vision research on visual-gestural language [68, 77]. Data from ASL native signers are collected with video collection equipment and users can enter annotation information into data distinct fields. The video clips and associated linguistic annotations are available in multiple formats for ASL studies and gesture analysis. Another example concerns the design of an online web browser for the deaf community [44]. It provides hyperlinks within video in a sign language-based, text optional web environment.

Today, lifelike virtual human figures can be constructed (examples can be found in [21, 23]). Human avatars can imitate human actions and even facial expressions. All the body joints and featured parts (such as eyebrows or mouth), represented as various parameters, are controlled in their motions, allowing the creation of virtual gestures.

The use of virtual human figures in sign language studies is ongoing [3, 4, 22, 50]. The main problems with these studies' results are that the users have to know English to utilize them and that there is no user control over the signing avatar. Therefore, the users cannot create new sign language "words" or "phrases." This is a significant limitation because natural language should be dynamic and extensible. The lack of easy-to-use interfaces also makes it difficult for beginners to benefit from the results of these studies.

In 1993 there was a panel discussion at ACM INTERCHI [45] where a distinguished panel of experts laid out a framework for what they felt would be a good interfacing system for sign language communication. The computers at that time were not capable of generating the real-time graphics that were required to implement such a system, and so work in this area stalled. We now return to what they proposed and are able to introduce new concepts into such an interfacing system.

Chapter 3

Overview of the Interfacing System

This chapter will briefly introduce the functionality, architecture, design philosophy and implementation of the proposed sign language interfacing system.

3.1 About the Sign Language Interfacing System

The proposed sign language interfacing system is an interactive application program which can run on various computer environments including different hardware configurations and software (operating) systems. The system, once started up, will provide the user suggestive graphical widgets and meaningful icons as input tools for giving input commands and controlling parameter values. The input devices are the computer keyboard and mouse. The outputs are a virtual human body and its motion sessions (virtual gestures and signings) in both displaying format (real time on-screen displaying and image recording on disk) and "text" patterns which are represented as a set of parameter values corresponding to the virtual animations and as a collection of "writing symbols" in a particular sign language associated with the user's inputs and choices. All these operations are combined in a graphical user interface (GUI) wrapper.

The ultimate goal of this system is to provide a working platform for any sign language user to (1) construct linguistic parts (such as phonemes and morphemes) for any sign languages, (2) create sign language vocabulary, (3) build up dictionaries for sign languages, and (4) “write” in a sign language. The users of the system can also produce any imitation sessions of the human body involving a wide range of human gestures.

First, the system has stored in its database frequently-used hand configurations and body postures from which specially-needed configurations and postures can be generated easily by fine tuning the rotation angles at body joints through various graphical widgets. These newly user-created representations can be put into the database for later use. The users then use a gesture editing GUI which allows for combining different selected body configurations together with their corresponding timings, the result of which is an animation session of the ordered gestures. The interpolation of interval body configurations between two adjacent body postures are automatic depending on the timing factors and joint positions of the adjacent postures. The interpolated interval postures can be edited dynamically with the adjustment of the body joint angles and actualized and inserted as a permanent gesture into the animation procedure. Natural and smooth transition from one posture to another one is therefore guaranteed. In this way, users can define, design, and construct sign language phonemes and morphemes and associate them with writing symbols in a certain sign language. These smaller linguistic parts, together with their writing representations, are stored in a database for the creation of larger linguistic parts such as words and phrases.

A sign language word consists of phonemes, morphemes, or appropriately-defined gesture sessions and can be constructed in a similar way to that described above. The only additions are the incorporation of gesture animations instead of only body postures and the introduction of related words in the word database management. The created words, their related words (such as synonyms and antonyms), and functions of incorporating two and more words into “phrases” and “sentences” are foundations for the construction of a sign language dictionary. Database design and management for the words are also an important issue.

Two main problems prevent “writing” in sign languages: how to retrieve in the shortest time a right sign (word) from the sign database and how to transition smoothly, naturally, and grammatically from one sign to the next following the grammatical rules of a sign language. To deal with the first problem, we can borrow methods such as *autocomplete* used in several Asian language text input techniques. When one clicks on a sign or types in the transcription code for a sign, the signs related to that sign (*e.g.*, with higher associated weights or sharing the first transcription coding symbols) will be displayed on the screen, each of which is an animation sequence accompanied by its writing representation and/or a number, rendered in an easy-to-understand style in a small screen area. One can easily click on the desired sign or type in its representative symbols or the number.

The second problem lies in the area of sign language linguistics. Our task in the interfacing system is to provide an efficient GUI tool for editing the transitional signing process between adjacent signs, such as phonological precesses, numeral incorporation and compounds. Without considering the linguistic meaning of a signing process, sign editing can be seen as deletion of interpolated interval body postures or the postures at the end of the first sign and at the beginning of the following sign, adjustment of these postures’ joint angles, and modification of the timing factors of these postures.

3.2 The Architecture of the Interfacing System

The sign language interfacing system provides an interactive virtual environment in which users can construct virtual gestures through a virtual human body, associate the gesture sessions with sign language linguistic parts (such as morphemes, words, and phrases), create virtual signs with the use of databases for managing the linguistic parts, and even "write" in a sign language.

The system has five main functional components: (1) a virtual human body, (2) a virtual environment, (3) inputs for the body control and environment setting, (4) outputs of the results in various formats, and (5) a database of the virtual gestures and signs. The architecture of the system is shown in Figure 3.1.

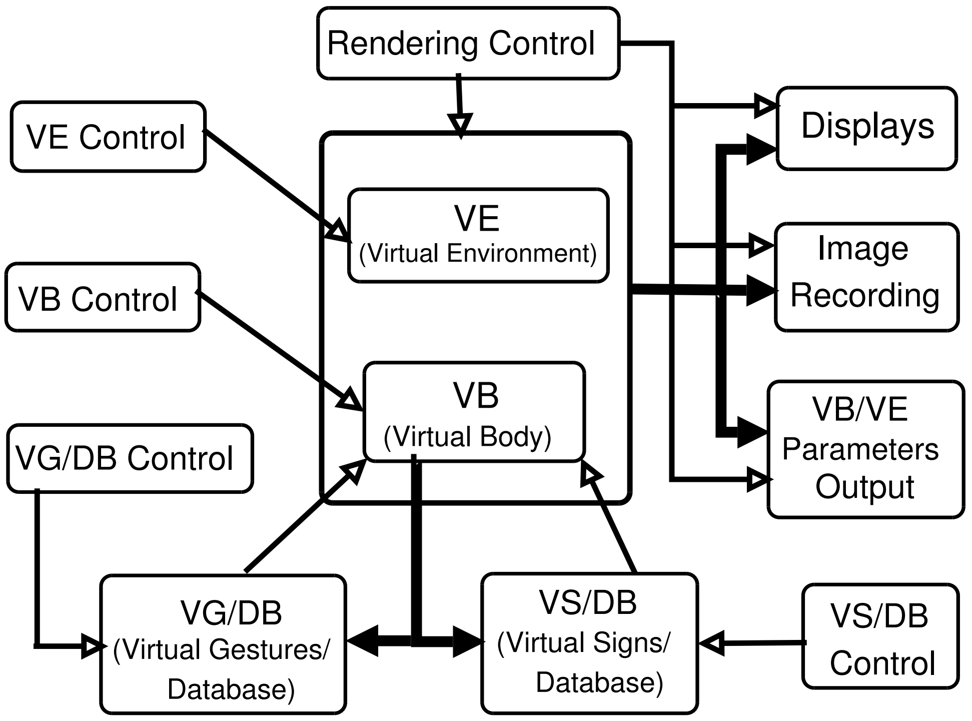


Figure 3.1: The architecture of the sign language interfacing system.

The virtual human body

The central and foundational part of the interfacing system is the virtual human body which displays the results by performing the operations from input commands or taking parametric values from the databases for its functional parametric parts.

The virtual human body is modeled based on the anatomical structure of the human body. A male body is used in the system. The whole body is divided into various functional parts, each of which is represented with a set of parameters. For example, the left forearm is one functional part of the body, and its parameters include the elbow joint location and rotational axes for the elbow bending and rotation (twisting). Simulation of basic facial expressions is also considered.

The virtual environment

The virtual body is placed in a virtual environment, which is virtual box. Several virtual cameras are “installed” in the box, and the background can be set to different colors. Because the hand plays the most important role in signing, two virtual settings are created for both hands of the virtual body.

The input/control part

The input/control part of the system is used to provide the inputs either from the databases or from users to the virtual body for producing body postures and gesture sessions, to set up a particular virtual environment (such as background and camera setups), and to control the output processes and formats. This part includes the following components:

- *VB Control*: to provide rotations at the body joints and define facial expressions.

- *VE Control*: to set up the virtual environment and fine-tune virtual cameras; to select and place a character as the virtual sign from a character pool (male and female virtual characters in diverse races).
- *Rendering Control*: to render the virtual body in a chosen style, to display and record the outputs (images and parameter values for the virtual body and environment) on the screen or disk drive in a certain format.
- *VG/DB Control*: to construct and edit virtual gestures and store/retrieve them to/from the virtual gesture database.
- *VS/DB Control*: to create and edit virtual signs and associate them with a particular sign language notation; to store to /retrieve from the virtual sign database the virtual signs and their corresponding/related notations.

The output part

The output of the system is represented in different formats: visual display on the screen, image session recording to the disk, and parametric representations of functional parts of the virtual body and virtual environment.

- *Displays*: The avatar making virtual signs on the screen in a specifically-set environment. There are two additional display windows exclusively for displaying the animations of both hands.
- *Image Recording*: Image frames of a virtual sign can be recorded on disk as separate image files in a certain format; these images can be combined to a movie.

- *VB/VE Parameters Output*: The virtual body is seen mathematically as an ordered set of parameters. A virtual gesture is a set of defined values for such a parametric set. These values can be recorded in a text file as an output of a virtual gesture. In addition, the relationship of the virtual body to the virtual environment, and the relationships among the functional body parts during their movement, can be represented mathematically as a set of motion matrices, all of which can also be recorded in a text file.

The gesture and sign database part

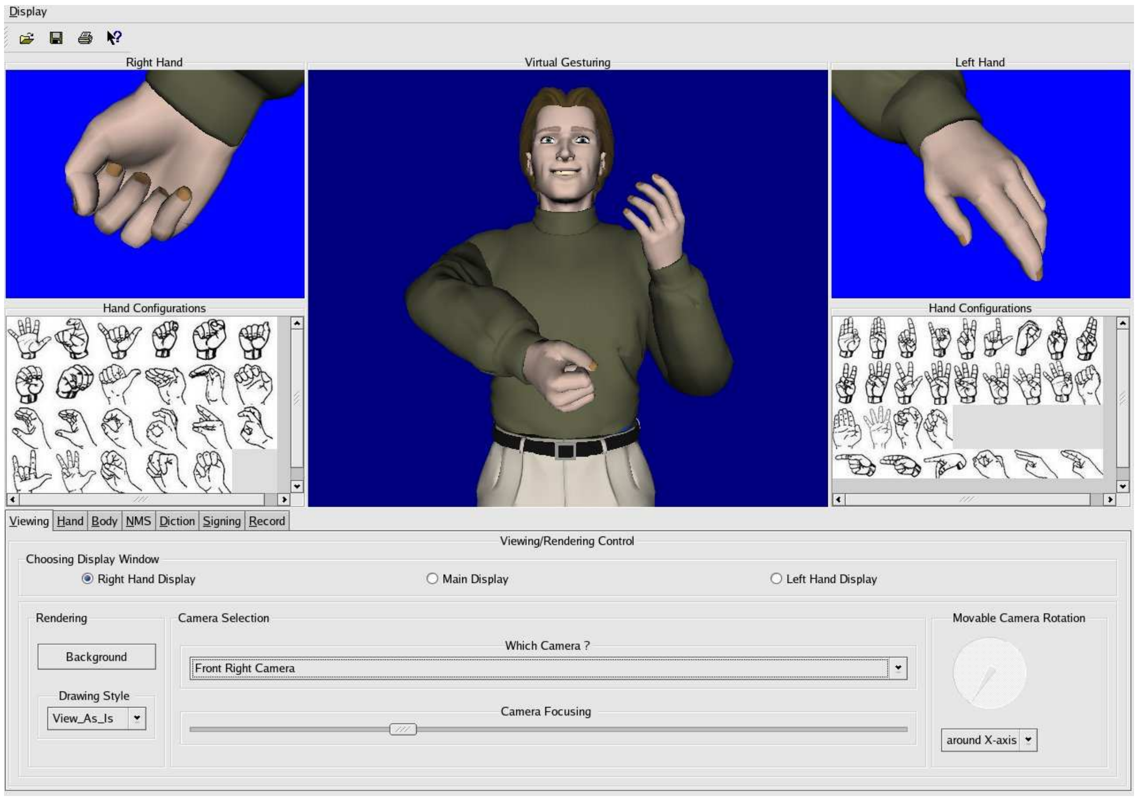
All the created gestures, virtual signs, and their sign notations are stored in their databases. Gestures and signs are sets of parametric values defined in particularly designed data structures. Sign notions at the current stage of development are English translations of the signs. Users use the database's creation and editing of new gestures and signs through the database control part.

- *VG/DB Database*: a database for virtual gestures. A gesture is a list of virtual postures with a timing factor for each of the postures. Thus a gesture, after being loaded from the database, is displayed as an animation session on the screen. This gesture database includes sub-databases for body postures and hand configurations.

- *VS/DB Database*: a database for virtual signs. A sign consists of one or more virtual gestures and is a sign linguistic part in a particular sign language. A sign is stored in the database together with its sign notation and the links to its related signs.

The system layout

One screen shot of the sign language interfacing system is shown in Figure 3.2 to illustrate the system's architecture. The upper part of the system layout is for display with a main display window in the middle for displaying body gestures/signs and two



[✓ Right Hand Display]

Drawing Style: View_As_Is

Which Camera ? Front Right Camera

around X-axis

Figure 3.2: One screen shot of the sign language interfacing system.

accessory (smaller) ones on either side for demonstrating hand gestures. Below the hand display windows are hand icons of the most frequently used hand configurations in sign languages. Clicking on a hand icon will load from the hand database the corresponding hand configuration to the current hand (right or left). The lower part is arranged as a set of graphical tabs for virtual body and environment control, gesture and sign creation and editing and their database management, and output rendering and recording.

3.3 The Design and Implementation of the Interfacing System

In this section, we briefly explain the general principles in developing the sign language interfacing system and then introduce computer system libraries and tools for the implementation of the system.

3.3.1 The System Design Philosophy

The sign language interfacing system is a complex project that involves the utilization of advanced computer graphics technologies (in modeling the human body), the application of software engineering principles (particularly that of usability consideration in the GUI design), and the exploitation of research on sign languages. The general methodology in designing the system is modular design in software development introduced in [33, 79, 89].

Modular design for a complex system means splitting the whole system into smaller parts (modules), each of which is well defined according to its functionality in the entire system and re-used and scaled up conveniently for future updating. We have applied this modularity to the interfacing system by:

- partitioning the system into discrete modules which are relatively independent and self-contained functional elements.
- making use of well-defined data types following the object-oriented descriptions of module functionality.
- streamlining the exchange of information among the system modules with carefully-defined module interfaces based on the information hiding consideration.

The architecture of the system (see Figure 3.1) can be used as a reference on how to divide the system into modules. Every component is a module that performs independently a portion of the system function. And a module can be partitioned into even smaller parts, which are also modules. For example, the virtual human body is one of the main components (modules) of the system, and it is further divided into several smaller parts such as the shoulder or head (see Figure 3.3a).

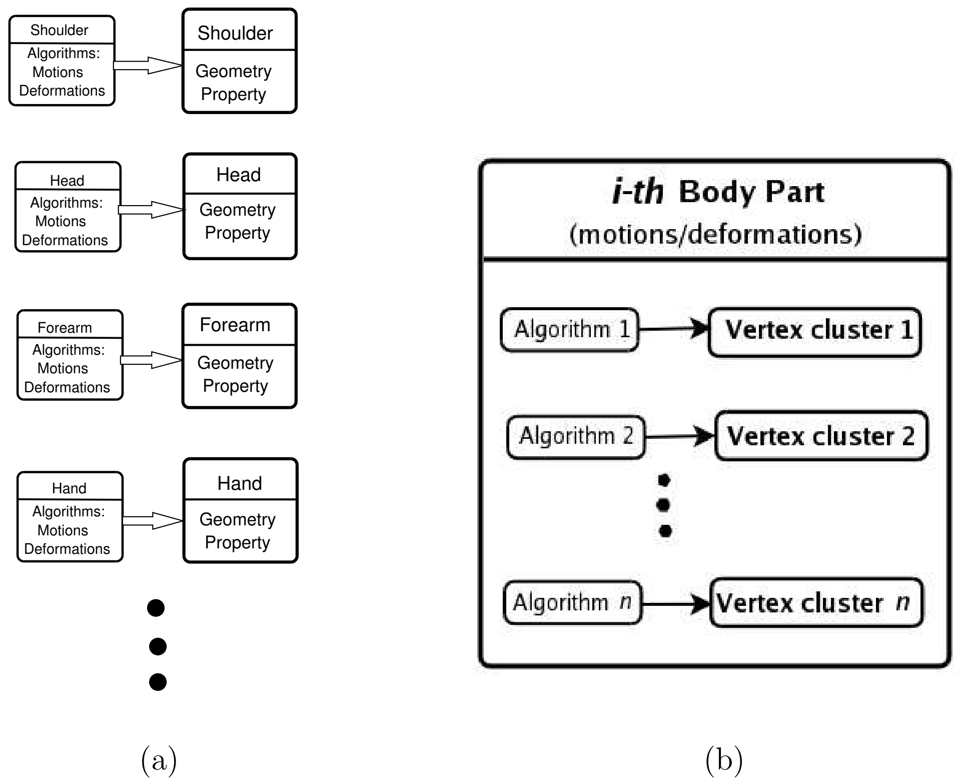


Figure 3.3: Partitioning of a module: (a) a human body is a module which can be further divided into even smaller modules (corresponding to smaller body parts); (b) these smaller modules share a common abstract structure, and they each can be implemented independently according to the the distinctive features of the smaller body parts.

One of the advantages in modular design is the convenience of subdividing a module into several sub-modules, all of which have individual characteristics but share common features to some degree. For example, a virtual human body consists of chest, hands, head, shoulders and other parts. We can use a common abstract data type to describe the structure of these smaller parts, as shown in Figure 3.3b. Any such a sub-module is made of n clusters of 3D vertices, and each cluster has its own motion pattern described by its specific algorithm. Some aspects of a sub-module can be predefined: (1) the relationship between each cluster and its algorithm, (2) how an algorithm is applied to its corresponding vertex cluster, and (3) default properties used when rendering the vertex cluster.

Thus the implementation process—writing computer programs to model these sub-modules—becomes quite simple, because all the component modules of a virtual body can use the same abstract structure. The remaining work in modeling a sub-module is to investigate characteristics of the sub-module and then decide: (1) how to split the sub-module's vertices into n clusters; (2) how to design a mathematical algorithm for each cluster; and (3) what properties will be used for rendering the cluster vertices.

Another advantage of a modular system lies in its ease of scalability, portability, and reusability. Figure 3.4 gives an example where a new human body model is used to replace the current avatar in the sign language interfacing system. This replacement requires the following: is: (1) determine the differences among relationships between the input/output requirements, formats, and domains of the two models; (2) design and implement input/output interfaces that overcome the differences and reflect the relationships; (3) combine the input interface, new body model, and output interface into the current system as illustrated in Figure 3.4.

The human body modeling module, input/control module (GUIs for user-avatar

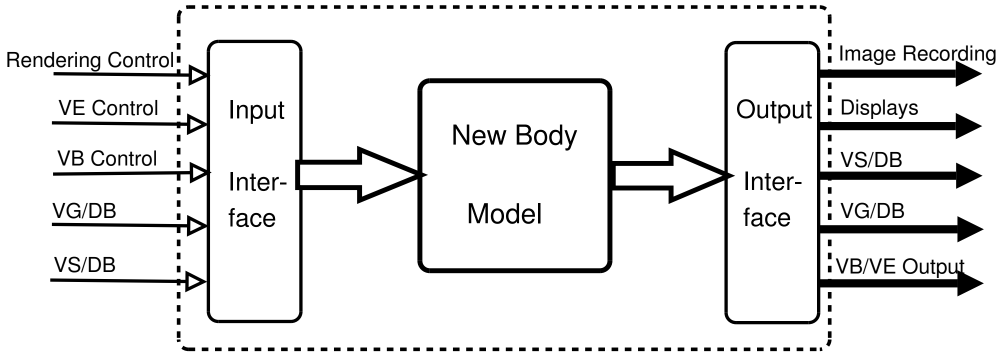


Figure 3.4: The substitution of a new body model for the old one.

communication), output module (such as image formats), and gesture/sign database module are implemented with different computer system tool-kits and libraries. These computer system programs may be updated at any time; and, with the consideration of modularity in our sign language interfacing system, only those modules implemented with the updated toolkits need to be revised.

3.3.2 The Implementation of the System

The sign language interfacing system was developed under the Linux (RedHat/Fedora 4) operating system on a PC with C/C++ and Perl [18] programming languages. The virtual body modeling is implemented with OpenGL [17] and OpenInventor (Coin3d [20]) graphical libraries, graphical user interfaces with Qt [19] tool-kit, gesture/sign databases with MySQL [16], and we used KDevelop [13] as an integrated development environment (IDE).

All these software toolkits and libraries have their open source versions under Linux and Windows. We have also tested the virtual hand simulation system on Windows.

Chapter 4

Modeling the Human Body

The virtual human body is a central part of the sign language interfacing system. This central component is seen as a module, which can be further divided into smaller modules (sub-modules) according to the hierarchical structure of of the human body as the body moves and the motions of its body parts must be coordinated. In this chapter, the structure of the body is first introduced, then a general body part model and its motion engine are presented, and finally modeling of specific body parts will be illustrated with the emphasis on hand modeling.

4.1 The Hierarchical Structure of the Human Body

The first step in human body modeling is the representation of the body—a complex organic structure of bones, joints, muscles, and other elements. For the study of its movements, a skeleton representation might be a good starting point. But to describe the gestures of the human body, a simplified "box" representation serves as a better illustration diagram. See Figure 4.1a, in which each "box" represents a major motion part of the human body.

The lower parts of the body (legs, feet, and hips) are rarely used by sign language users during signing, and thus the lower limbs are simplified with a single box. We use the *Torso* to represent the trunk of the body, which is connected to the shoulders and head. The *Head* contains the eyes, hair, neck, and the frontal part, which is used to model the facial expressions. The *Shoulder* consists of the upper arm and the collar. The *Hand*, which is made of the palm, fingers and thumb, is the most important part of the body in the simulation of signing and is modeled differently than the other body parts. The *Forearm* connects the shoulder (upper arm) and the hand.

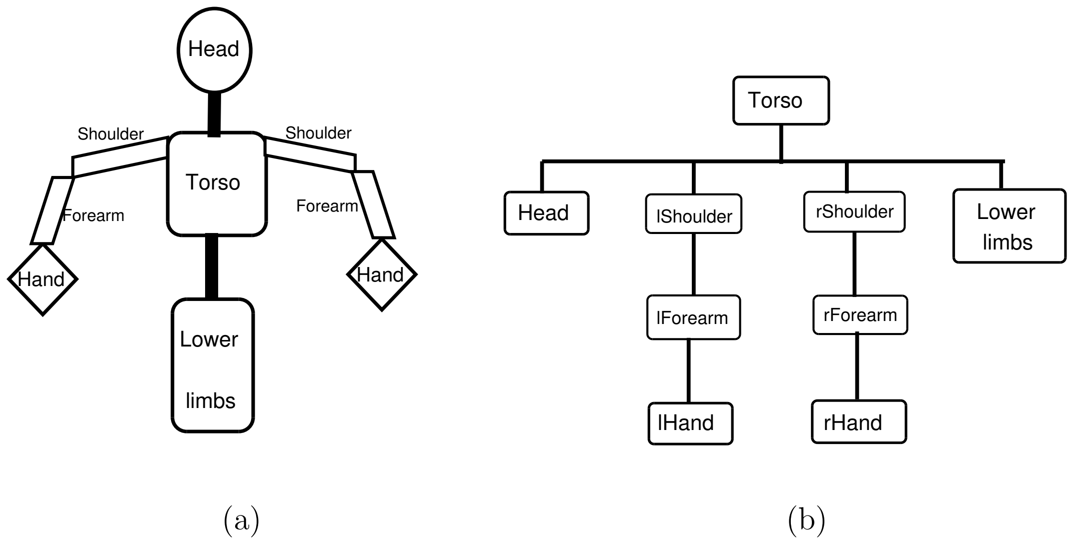


Figure 4.1: Representations of the human body: (a) a simple diagram representation for the human body; (b) a tree structure of the human body.

A tree structure is often used to design algorithms for the movements of the objects that are connected in some order as follows: the movements of a parent node will automatically propagate to all of its child nodes (children); the ultimate movements of a child are the cumulated combination (multiplication in mathematics) of the movements of all its parents in the tree. A human body in making signs can be represented with a tree structure illustrated in Figure 4.1b in which the nodes have their corresponding "boxes" in Figure 4.1a (*r* and *l* indicating the right and left body parts).

The movement of the lower limbs (*e.g.*, walking) is interpreted as "displacement" of the torso, thus making the torso the father node of the whole tree. If the torso moves, all the other body parts (head, shoulders, lower limbs, *etc.*) will respond with follow-up movements. For example, *lForearm*'s movement not only includes its own motions (bending and twist) but also depends on the movement of *lShoulder* and that of *Torso*.

The hand is divided into a palm, four fingers and a thumb; a finger is further split into three finger parts; and all the hand parts are connected in a more complicated tree structure. It seems a very complicated task to describe mathematically the movements of some nodes, but things become clear and simple if we assign a local coordinate system for each of the nodes in the tree structure, consider a node's movement only at its local coordinate system, and transform the tree structure of body parts into a tree structure of their corresponding coordinate systems, as Figure 4.2 illustrates.

Furthermore, we introduce homogeneous coordinate representation to simplify the calculation of the node movement. Here, an object's movement, both rotations (in any direction around any point) and displacement (translation along any direction) can be described using a 4×4 matrix; multiplication of matrices is the only operation in the calculation of combined movements; and you can tell the difference between a point and a vector by their mathematical representations.

For example, *rForearm*'s movement at its local coordinate system can be described with a 4×4 matrix $M_{rForearm-local}$; the relationship between coordinate system *rForearm* and coordinate system *rShoulder* with a 4×4 matrix $M_{rForearm-rShoulder}$; the relationship between coordinate system *rShoulder* and coordinate system *Torso*

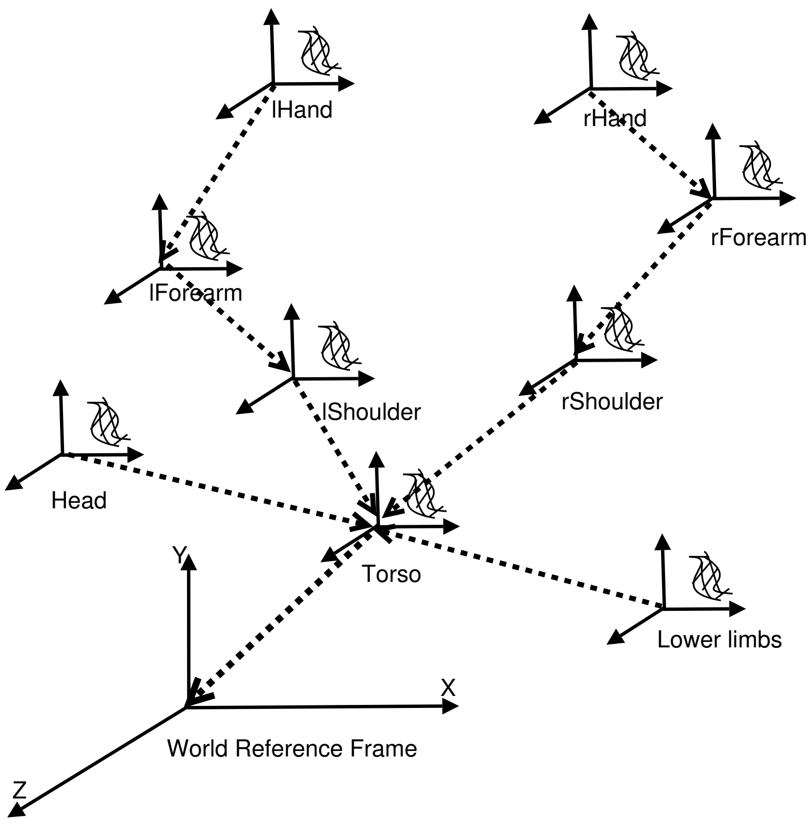


Figure 4.2: Coordination of the motions of the human body parts.

with a 4×4 matrix $M_{rShoulder-Torso}$; and the relationship between coordinate system *Torso* and coordinate system *World Reference Frame* with a 4×4 matrix $M_{Torso-World}$. Thus, *rForearm*'s movement in the *World Reference Frame*, $M_{rForearm-World} = M_{Torso-World} \times M_{rShoulder-Torso} \times M_{rForearm-rShoulder} \times M_{rForearm-local}$. Order in matrix multiplication is important. The above formula must be read *from right to left*, that is, from the right forearm to the right shoulder to the torso, and finally, to the *World Reference Frame.*

The introduction of the local coordinate system aids the modular design for specific body parts, particularly the design of deformation algorithms for the part.

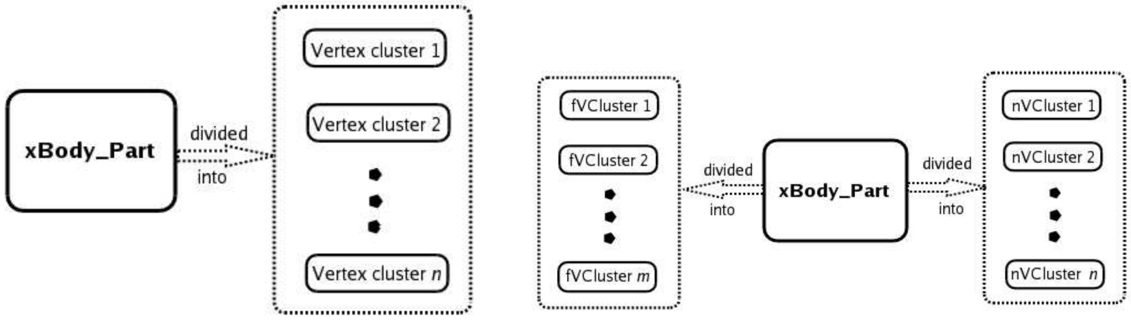
The influences of other adjacent body parts on the deformation of the local part and the influences of the local part on the other adjacent parts are communicated by user-defined data types. Issues of deformation analysis and simulation are kept only in the local coordinate system.

Using relationships among the coordinate systems instead of those among the body parts *per se* has greatly reduced the work in programming with graphical libraries in modeling the human body and simulating human gestures. There are some movements in a body part that are not propagated to the nodes in the tree structure, for example, initial adjustment and deformation of a body part. For the movements of a body part that should be conveyed to other parts (*e.g.*, raising *rShoulder* has influence on *rForearm* and *rHand*), the motion matrix of that body part is simply assigned with these movements. Most importantly, we can take advantage of some features of the graphical libraries so that the computer rendering mechanism can automatically update the movement of a coordinate system to its child coordinate systems.

4.2 A General Body Part Module

All the body parts that make up the human body share some common features, and when modeling them in a particular computer programming language, these features take the form of user-defined data types, functions, and rendering algorithms for the modeling implementation. We can define an abstract body part that accomplishes all operations for these common features.

A body part is geometrically a set of 3D points. Without the influence from other body parts, as Figure 4.3a shows, a body part (*xBody_Part* in the figure) is divided into n smaller parts, each of which has its own deformation method. However, most body parts are as illustrated in Figure 4.3b, where a body part is influenced by other parts and exerts influence on other parts. The former case occurs when a body part's immediate child, influenced by a motion (such as bending and/or rotation), changes the shape of the body part (the closer to the child, the larger the deformation). The latter happens when a body part is in motion, and it affects its parent part by changing the parent's shape. For example, suppose *rForearm* is the body part in question. When it bends *rForearm* not only changes its shape but also its parent's (*rShoulder*) shape in places around the border between them. And as *rForearm*'s immediate child, *rHand* (the right palm, to be exact), bends or moves side-to-side, and *rForearm* will deform around the border between *rForearm* and *rHand*.



(a) (b)

Figure 4.3: Modeling of a human body part with deformation consideration: (a) deformation on only one end; (b) deformation on both ends.

By investigating human body parts, we conclude that a more general body part should have the characteristics demonstrated in Figure 4.3b and that the type described in Figure 4.3a is a special case. Now, the question becomes how to describe and define, qualitatively and quantitatively, the movement patterns of a body part which can exert influence on, and be influenced by, others. We use Figure 4.4 to illustrate the algorithm in solving this problem.

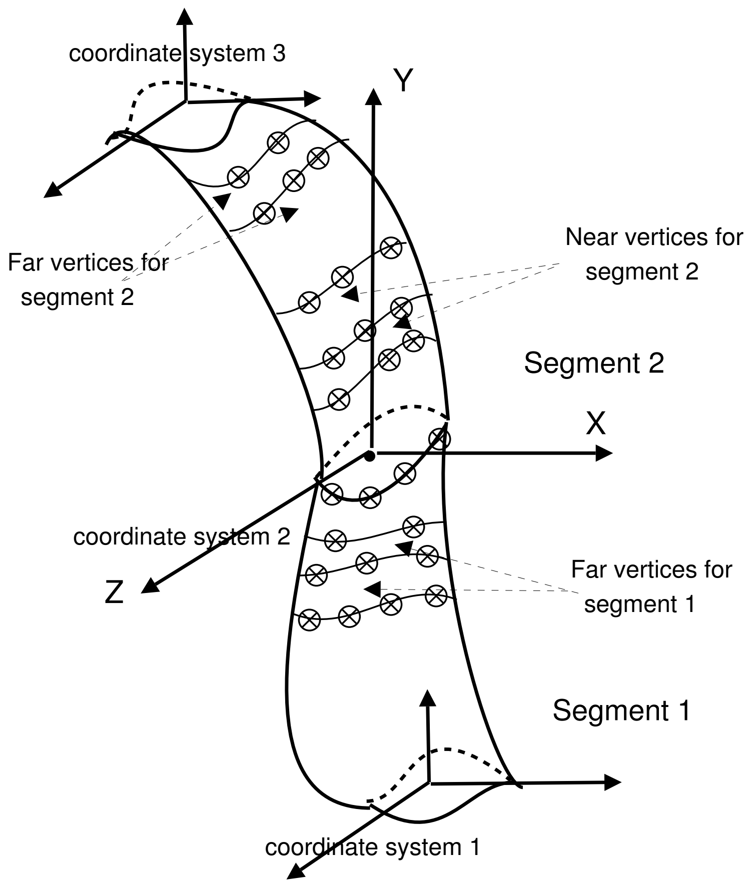

coordinate system 3
Y
Near vertices for segment 2
Far vertices for segment 2
Segment 2
X
coordinate system 2
Z
Far vertices for segment 1
Segment 1
coordinate system 1

Figure 4.4: Definition of *near* and *far* vertices.

There are three parts chained in order: *Segment2* is the body part we have interest in; *Segment1* is its father in their tree structure, and a body part in *coordinate system 3* is its immediate child. When *Segment2* rotates, we consider three motion patterns in the simulation of the body part's rotation: (1) *Segment2* is seen as a rigid body, and thus all its points have the same rotation as the body part does, and this rigid rotation will propagate to *coordinate system 3*—because in a tree a father's movement should transfer to its immediate child; (2) some points that are close to the origin of *coordinate system 2*—(near vertices of segment2) will have deformation in addition to their rigid rotation with the deformation algorithm determined by the body part's mechanism; and (3) *Segment2*'s rotation will produce deformation on the points (in *Segment1*) that are close to *Segment2* (the points called *far vertices* of *segment1*). We don't consider these points because at the moment we are modeling only *Segment2*. However, we do consider *far vertices for segment2*, which will deform under the influence of the part that resides in *coordinate system 3* (*Segment2*'s child).

Having explained a body part's relationship with its immediate child and its father, we model the body part with consideration of its deformation. First, we define two coordinate systems for the part (see Figure 4.5): *fCoord* (far coordinate

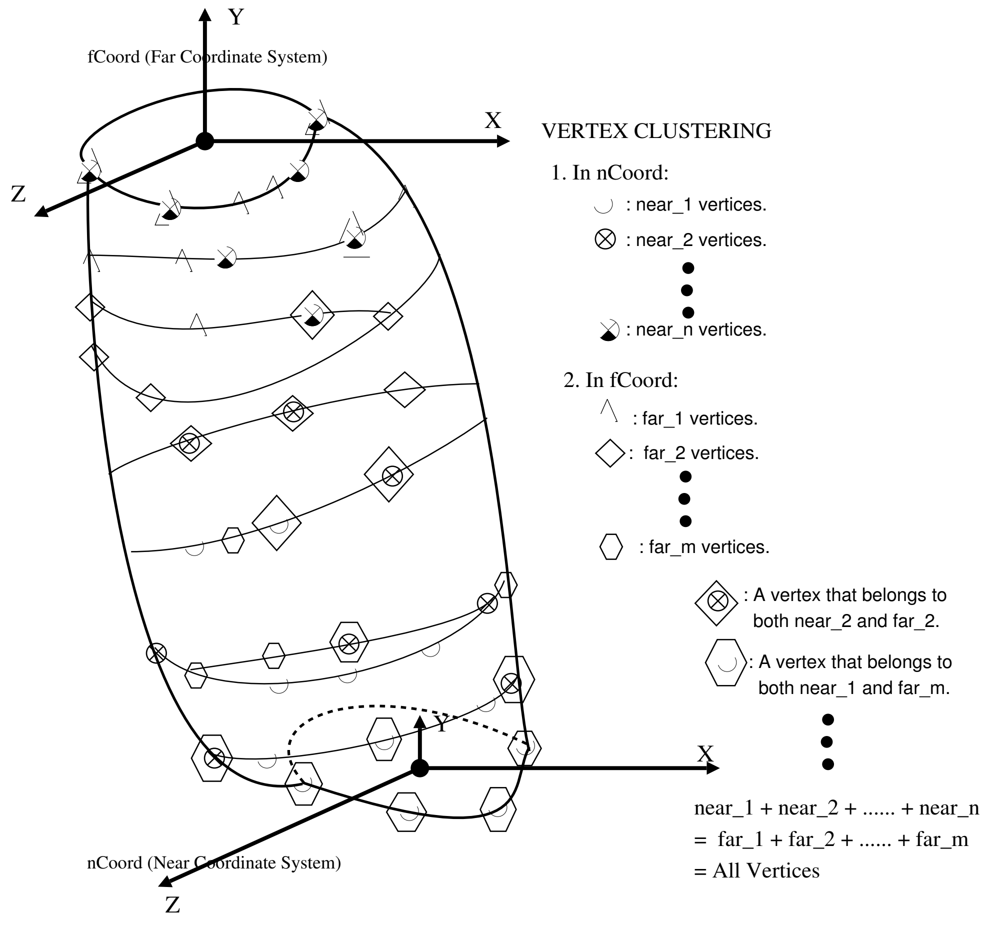

Y
fCoord (Far Coordinate System)
X
Z
VERTEX CLUSTERING
1. In nCoord:
: near_1 vertices.
: near_2 vertices.
: near_n vertices.
2. In fCoord:
: far_1 vertices.
: far_2 vertices.
: far_m vertices.
: A vertex that belongs to both near_2 and far_2.
: A vertex that belongs to both near_1 and far_m.
Y
X
Z
nCoord (Near Coordinate System)
near_1 + near_2 + ... + near_n
= far_1 + far_2 + ... + far_m
= All Vertices

Figure 4.5: Division of vertices of a body part into *near* vertex clusters and *far* vertex clusters.

system) and *nCoord* (near coordinate system). Then we split the body vertices in two types of vertex clusters: n near-vertex clusters and m far-vertex clusters. The definitions of coordinate systems and division of part vertices should comply with following rules:

1. each vertex belongs to only one near-vertex cluster and only one far-vertex cluster; the number of all the near-vertex clusters, the number of all the far-vertex clusters, and the number of all the body parts are the same.

2. *nCoord* is defined by the body part and its father; *fCoord* is defined by the body part and its immediate child.

3. *nCoord* and all n near-vertex clusters are used to model the deformation resulted from the body part's movements.

4. *fCoord* and all m far-vertex clusters are used to model the deformation resulting from the movements of the part's immediate child.

With the above description of a body part, we can now define a general body part module in an object-oriented programming style: data types and their dynamic memory assignment mechanisms for *fCoord* and *nCoord* (their origins and rotational axes and points), near-vertex clusters, far-vertex clusters, movement matrices, the implementation mechanism for applying deformation algorithms (the algorithms are produced by a motion engine, as discussed in the following section), all the vertex properties and their geometrics, and the rendering mechanism for the vertices.

4.3 A General Motion Engine

As described in the previous section, a body part is modeled according to its updated body vertices, and the body part will influence and be influenced by others connected to it, depending on the relationship between them. The general body part module does not know how its vertices are updated but only has an interface for importing the updated vertices. In this section, we will concentrate on a general motion engine that exports clusters of vertices and their properties (such as their normals), given a user-defined deformation algorithm for these clusters of vertices. When a motor engine is connected to a body part module, the rendering mechanism in the body part module will automatically (dynamically) render the body part.

Figure 4.6 is a design diagram for such a general motion engine. Because a motor engine is always used with a body part module, both of them have the same data structures in representing near-vertex clusters, far-vertex clusters, motion parameters (such as rotation centers and axes), and coordinate system parameters for the two vertex clusters. Users define the deformation algorithms for these vertex clusters by defining *deformation factors* for them. In the current version of our sign language interfacing system, a heuristic method has been applied to decide on these factors for different body parts.

When a body part moves or other parts connected to the part move, different types of signals will be sent from this part to the motion engine connected to the part. These signals will ignite a set of various calculations of new vertex positions, normals, and properties, depending on the types of the signals. These updated vertex data will feed back immediately to the connected body part module, which will start rendering the body part. The outputs also include motion information of the local

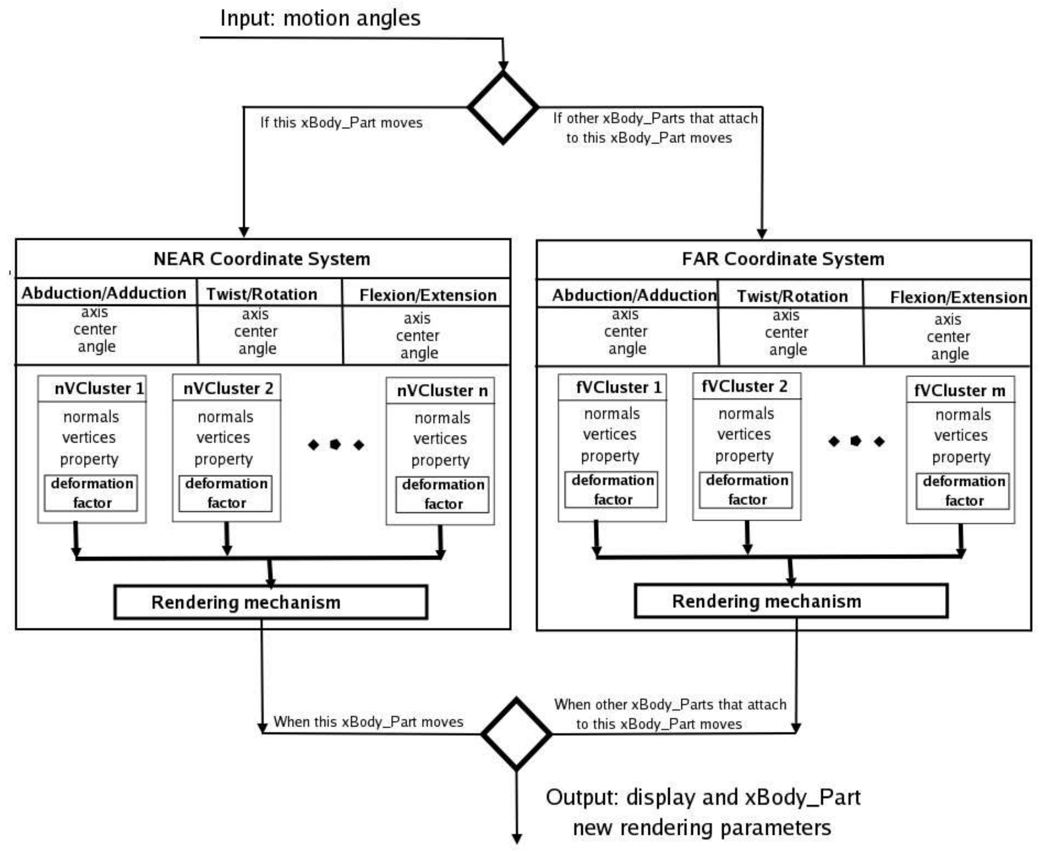


Figure 4.6: A general motion engine.

coordinate system for the body part, which will spread to the body part's immediate child, thus making all the children of the body part automatically render themselves.

We have described a general body part module and a general motion engine that can provide a body part its updated data for rendering. In the following sections we will illustrate how to use these two general modules to model specific human body parts, such as shoulders and upper forearms, forearms, the head with different facial expressions, and most importantly, the hand and its components.

4.4 Hand Modeling

In the review of research on the human hand in Chapter 2, we used Figure 2.9b to illustrate the motions of the palm, fingers, and thumb around the hand joints. Now with the introduction of the tree structure of the human body, a general body part module, and a general motion engine, we apply these concepts in the modeling of the human hand and the simulation of the hand motions.

There are sixteen hand parts in a hand: a palm, three finger parts on each of the four fingers (the index, middle, ring, and pinky), and three thumb parts. All these sixteen parts have an ordered tree structure as shown in Figure 4.7.

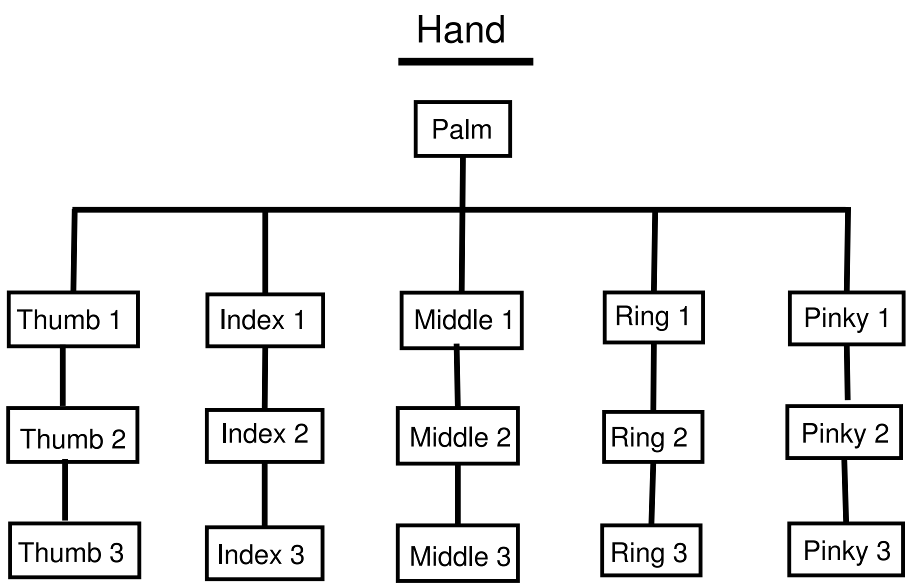


Figure 4.7: The tree structure of the hand.

Each node in the hand tree can be described with a general body part module, a general motion engine, and a local coordinate system. When hand parts move, there are movement signals exchanged between the adjacent nodes (hand parts) in the tree structure. To model a hand part, we feed the part's geometric data, motion patterns, rendering properties, and adjacent hand parts into its engine and modeling module. The engine also needs the deformation factors. Because a whole hand is often seen as one organized body part, we will in the following subsection describe the modeling process of the whole hand and the implementation of the hand constraints in the hand model, instead of depicting the hand parts individually.

4.4.1 Modeling Hand Parts

The starting point to create a hand model is to get raw hand data of reasonable size. Usually the data include the 3D positions of the points that define the hand and the connections among the points. While some companies provide hand model data with various resolution (*i.e.*, size), other sets can be found on the Internet for free.

Once the data is obtained, normals for vertices or polygons are usually calculated, although in some cases there is no need to provide normals because the rendering mechanism can automatically bind the vertices with corresponding normals. However, for the preparation of dynamic modeling where the hand is not considered as a whole object but as a combination of different rigid objects (fingers, palm), it is convenient to calculate normals at the beginning. In Open Inventor, we use *property nodes* to represent (define) appearance and qualitative characteristics of a scene (here, hand visualization). The hand 3D point locations, connections, and normals provide the *metrics* part of the Open Inventor *property nodes* for the hand model.

The *appearance* part of the property node is based on the *color value* given in each vertex. As for the hand, only two different colors are needed: one for fingernails, the other for the rest of the hand. The *transform* part of the property node is required if we want to manipulate (translate, rotate, scale, *etc.*) the hand in its display window.

With the 3D hand data we can reconstruct the hand model and render it. Figure 4.8 shows the reconstruction of a hand model. The forearm is included because movements at the wrist are an inevitable part of hand motion. Furthermore, without the forearm, the overall displaying effect will be awkward and uncomfortable to look at.

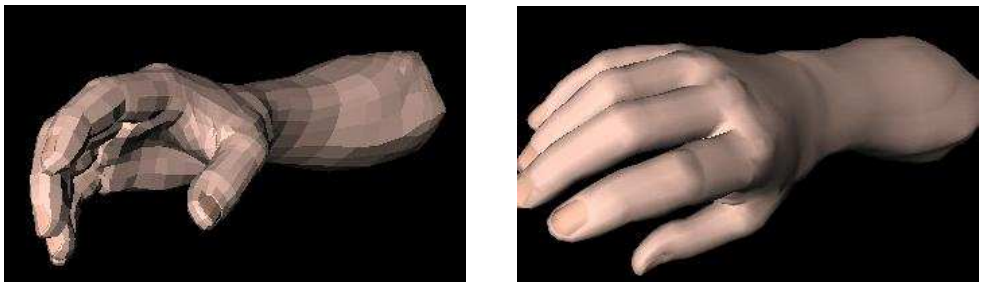

(a) (b)

Figure 4.8: A hand model: (a) the original hand, and (b) with normals applied.

The next step is to partition the original hand data by enumerating vertices and render them with appropriate styles. We then collect the possible edge vertices that share the border of two adjacent hand parts. Now we can render any hand part or any combination of some hand parts. Figure 4.9 shows an index finger, and Figure 4.10 shows a thumb base with all vertices numbered. In this way we can split the whole hand into sixteen hand parts as show in Figure 4.11.

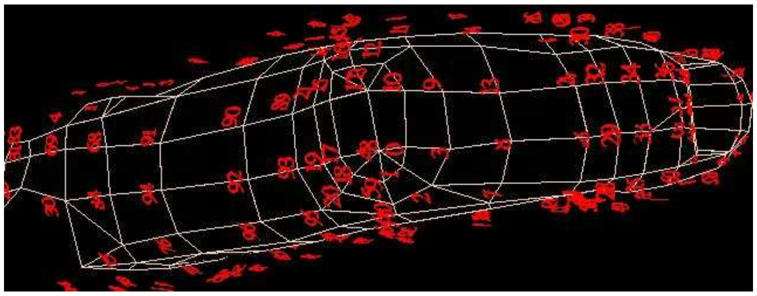


Figure 4.9: Index finger vertices marked with numbers.

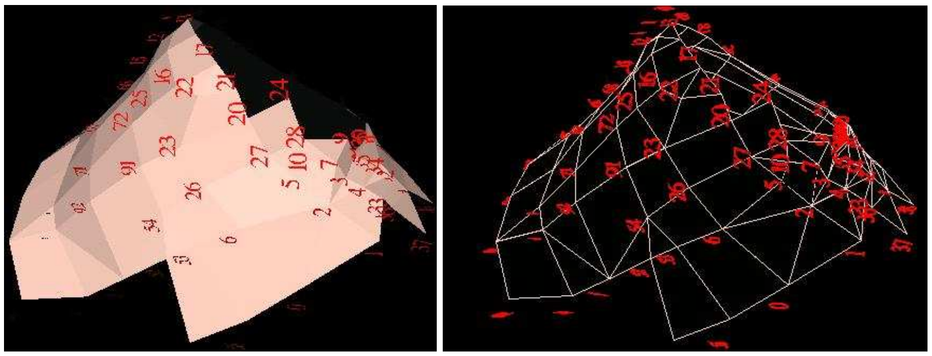

(a) (b)

Figure 4.10: Thumb base displayed in: (a) polygons, and (b) hidden lines.

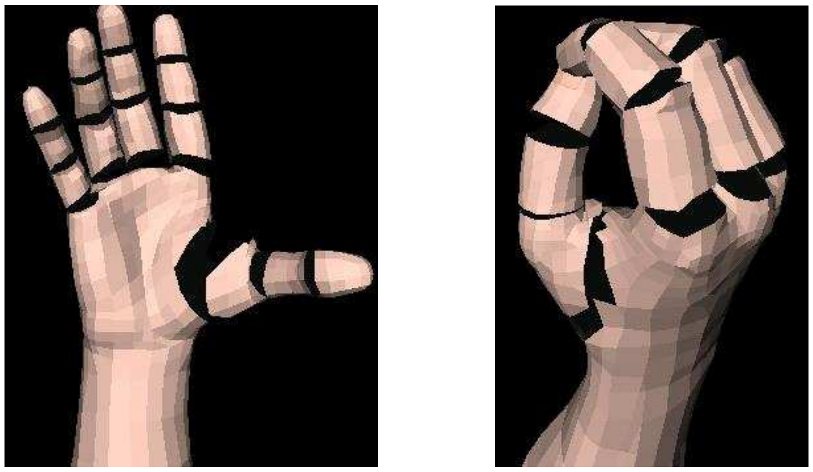


Figure 4.11: Partition of the hand.

Now we summarize the hand modeling steps. First, sixteen hand parts are identified: three for the thumb and three for each of the fingers and one for the palm. Then, the hand joints (finger's DIP, PIP, and MCP, thumb's IP, MCP, and CMC, and wrist joint) are calculated from the boundaries. Finally, a coordinate system is attached to each of the 16 joints. These coordinate systems serve as the object (local) coordinate systems for each hand part. One hand part belongs to each coordinate system and at the initial step is aligned with that coordinate system. Each coordinate system is attached to another coordinate system just as its corresponding hand part is attached to another hand part. For example, the index finger has three hand parts in their corresponding coordinates systems with origins at the index finger's DIP, PIP, and MCP joints. The coordinate system at the DIP joint is attached to that at the PIP joint, which is attached to that at the MCP joint.

Now we combine the anatomical characteristics of the human hand into our hand model. The anatomical features of the hand are maintained by keeping the length of any part's bone, that is, the displacement (distance) of two adjacent coordinate systems constant during hand motions. The center of the boundary of two adjacent hand parts is used as the origin of coordinate system. This approach is similar to the method for measuring hand motions used by medical and biomechanical professionals [39]. There is flexion at all the hand joints, and abduction at each finger's MCP joint, each thumb's MCP and CMC joints, and at the wrist joint. Other movements are added to the wrist joint: a rotation from the forearm's rotation and 3D translation as a result of the shoulder's motion. The hand parts are primarily rigid with the exception of the components around the boundaries of two adjacent hand parts. When a hand part rotates at a joint, the mesh points that are close to the hand joint on both adjacent hand parts produce the most deformations. Deformation weights are distributed according to point closeness and position.

Figure 4.12 shows several hand configurations rendered from the hand model with deformation considered.

4.4.2 Generation of Natural Hand Configurations

The hand, when in motion under constraints, generates *natural* hand gestures. One example of a natural gesture is that when you bend your middle finger at its MCP joint: your index and ring fingers will automatically follow the middle finger's

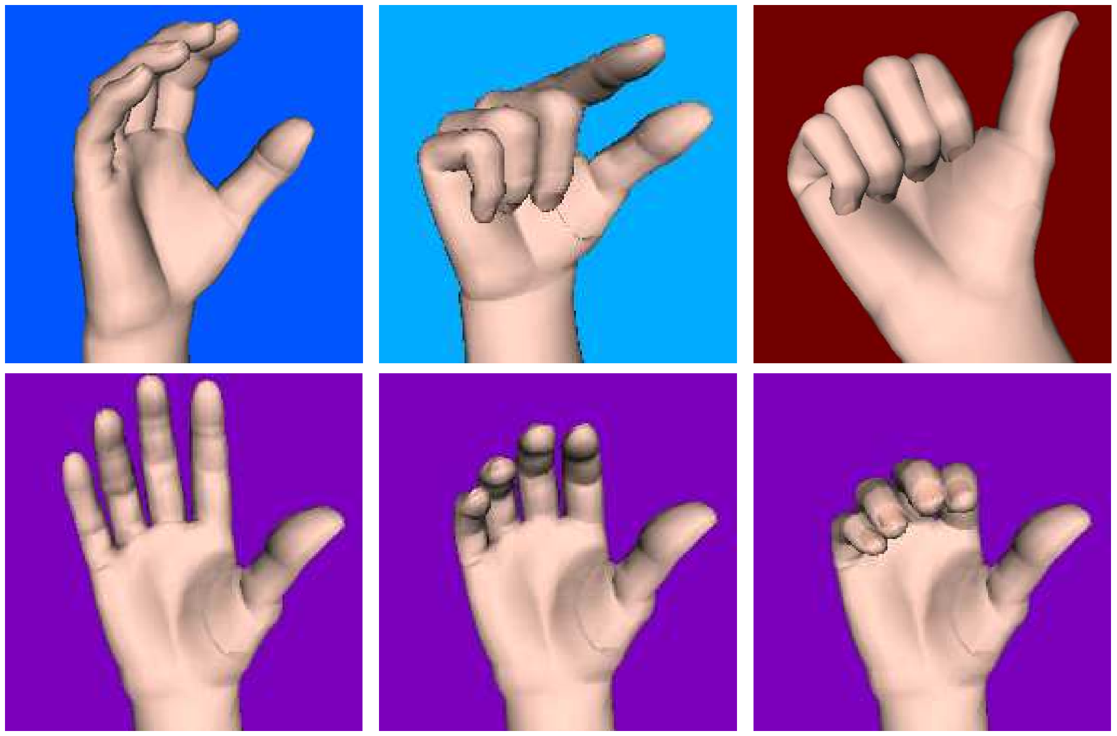


Figure 4.12: Several hand configurations rendered from the hand model.

movement and even your pinky will to a lesser degree. Although people display differences in the magnitude of hand constraints, all the natural movements observe the same pattern: the related fingers move the same way, only to different degrees.

The application of the hand constraints in our model is based on the constraint values obtained by medical researchers [39, 78]. First, the static hand constraints are embedded into the hand model, setting up motion (rotation) ranges for hand joint movements. Specifically, there are limitations on flexions at all the hand joints, on abductions at each finger's MCP joint, the thumb's MCP and CMC joints, and at the wrist, and on the rotation of the hand at the wrist.

The intra-finger dynamic constraints (how a hand joint in one finger affects other joints in the same finger) are applied by following the principle of two-thirds: the flexion angle at the DIP joint is two thirds that at the PIP joint, and *vice versa.*

Test and correction in the implementation of this principle on the hand model was necessary. Finally, the inter-finger dynamic constraints at the fingers' MCP joints (how one finger moving at its MCP joint affects the other fingers) is implemented.

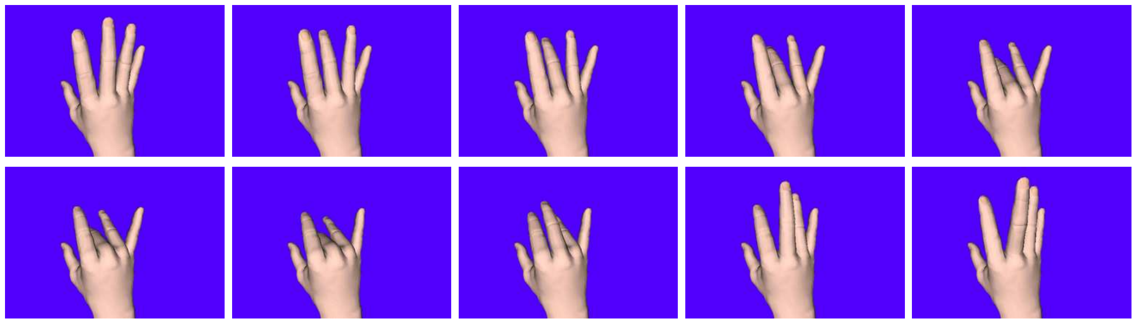
In the case of abductions, the middle finger can have slight abduction-adduction, and the ring finger follows the pinky when the pinky moves far away from the ring in abduction. The flexions at the fingers' MCP joints concerns with the following issues:

- When a finger (the active finger) bends at its MCP joint, it exerts influence on the MCP flexions of other fingers (the passive fingers). The influence varies depending on the active finger's flexion angle value (which of the low, middle, and high flexion subranges it is in) and on the closeness of the passive finger to the active finger. For example, when bending at the MCP joint, the index finger has the strongest influence on the middle finger, the moderate on the ring finger, and the least on the pinky. The influence becomes stronger as the index finger bends from the low subrange to the high subrange.

- The influence factors are not measured in a multiplying factor (by which the active flexion angle is multiplied to give the passive flexion angles) but are included in the maximum difference between the active flexion angle and the passive flexion angle. If the flexion angle difference between the active flexion angle and the passive flexion angle is larger than the maximum value, the passive flexion angle should be adjusted so that the difference is equal to the influence factor (the maximum value). For example, suppose that the index finger (active finger) bends to θ_I (active flexion angle) in one of the subranges and that the middle and ring fingers (two passive fingers) are at θ_M and θ_R. Also, suppose that the index finger in this subrange has an influence factor of 20^o on the middle finger and an influence factor of 35^o on the ring finger. There should exist the inequalities: $\theta_M \leq \theta_I \pm 20^o$ and $\theta_R \leq \theta_I \pm 35^o$. If either inequality is not satisfied, we adjust θ_M or θ_R such that $\theta_M = \theta_I \pm 20^o$ or $\theta_R = \theta_I \pm 35^o$. The positions of the active finger and the passive finger determine the choice of the sign "+" or "−".

- All the influence factors among all four fingers are estimated through experiments on our hand model and on real human hands.

Figure 4.13 shows some examples of the application of hand constraints at different finger joints: (1) bending and extending the middle finger at its MCP joint (the base middle joint) will result in the automatic follow-up bending movements of other fingers (the index, ring, and even the pinky finger); (2) bending the index finger at its DIP joint (the tip index joint) will result in automatic follow-up bending movements at its PIP joint (the middle joint on the index finger).


(a). Bending and extending the middle finger at the MCP joint.

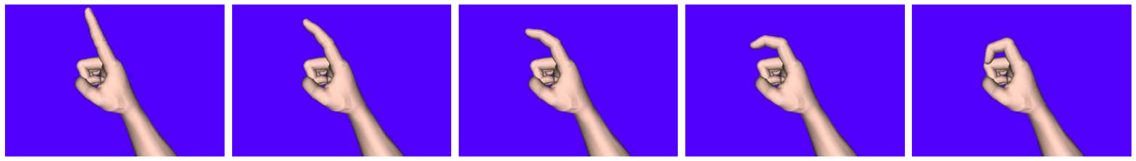

(b). Bending the index finger at the DIP joint.

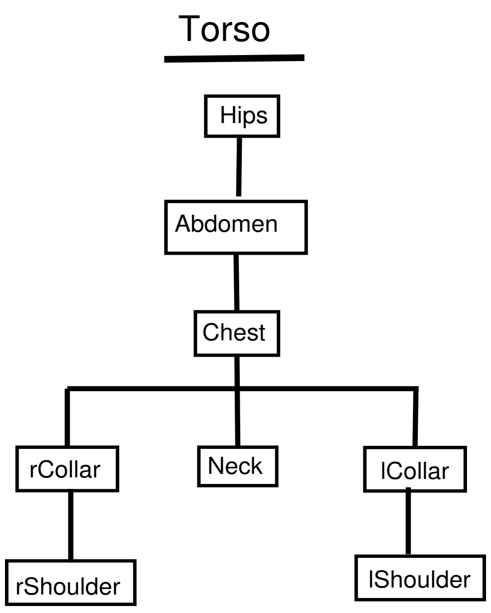
Figure 4.13: Application of hand constraints at finger joints.

4.5 Modeling the Other Body Parts

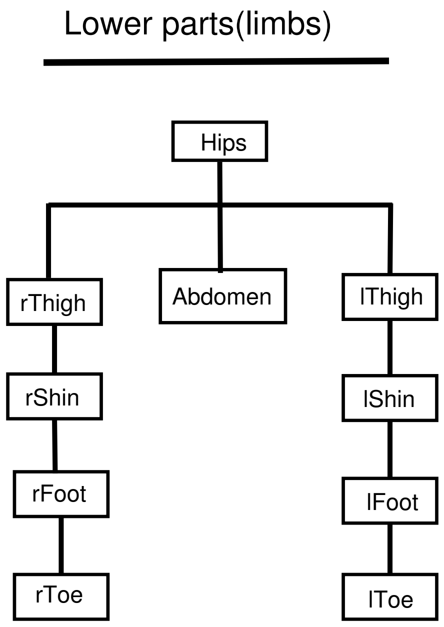
The simulation of the natural movements of other body parts (such as facial expression) is a challenging topic, and in our sign language interfacing system we target approximation in the simulation of body movements. A virtual human figure in the interfacing system is not a naked body, but one with clothes on, which results in an assumption for modeling the body parts—if a body part is *completely covered* by the clothing, then only the part of the clothing that covers the body part will be modeled. This will make the modeling process simpler and movement simulation closer to the body's natural motion. In this section we will first discuss modeling of the upper and lower body parts and then deal with the facial expression simulations.

4.5.1 Modeling the Upper and Lower Parts of the Body

The upper body consists of the abdomen, chest, collars, shoulders, and part of the hips and neck. All of these parts are covered by the clothing, and their movement patterns can be described with the tree structure shown in Figure 4.14a. All the nodes in the tree should be an independent body part module with an individual motion paradigm, but in our interfacing system we combine the hips, abdomen, chest, neck and collars into one large body part, which is a rigid body part with deformation only on the borders with the shoulders. The right and left shoulders, together with their corresponding upper arms, are modeled separately. We also consider the forearms in the upper body parts, though they are not included in the tree structure in Figure 4.14a, because we want to concentrate on the torso and its movements. Thus, we have five upper body parts to be modeled individually: the main part of the torso (the hips, abdomen, chest and neck), right and left shoulders (including the upper



(a)



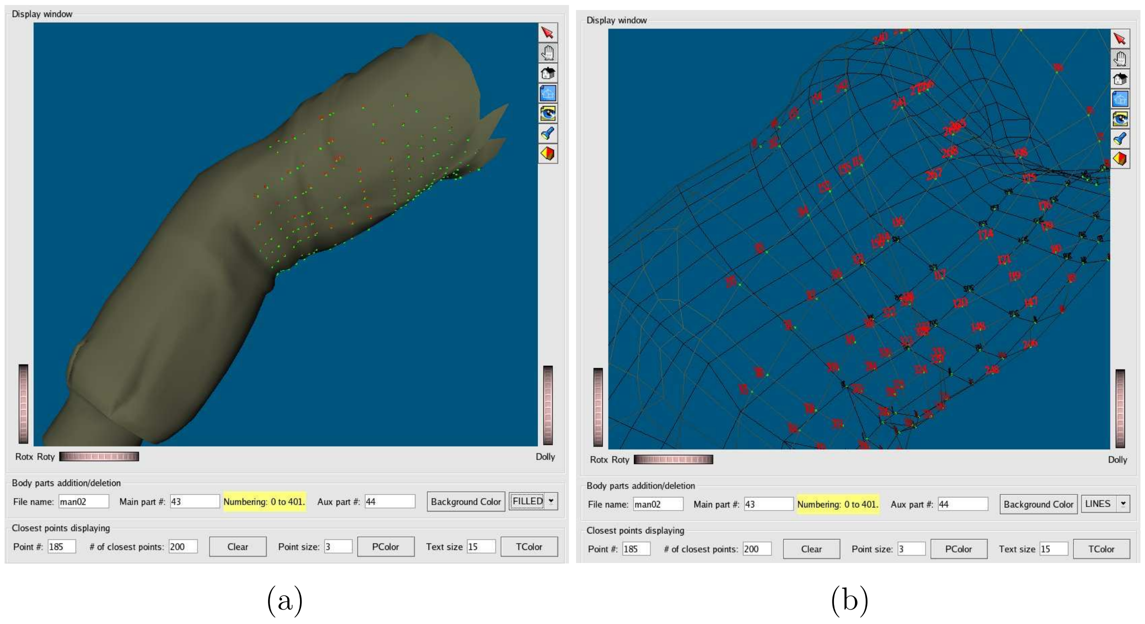
(b)

Figure 4.14: The tree structures of the upper and lower parts: (a) the upper part; (b) the lower part.

arms), and right and left forearms (attached to the right and left shoulders in the tree).

To investigate the motion patterns of the shoulders and forearms and design their deformation algorithm, we have implemented a tool (see Figure 4.15 for locating points of a particular body part). We can also use this tool to change the locations of the particular vertex clusters and then test the results.

By controlling the movements of some points on the shoulders and forearms and examining the results, we have found the points that can be used to calculate the movement parameters of these body parts, such as rotation joints and axes along different directions. In a similar way, we can design and test the deformation algorithms for the simulation of the shoulder and forearm movements.



Body parts addition/deletion
File name: man02 Main part #: 43 Numbering: 0 to 401. Aux part #: 44 FILLED
Closest points displaying
Point #: 185 # of closest points: 200 Point size: 3 Text size 15

Body parts addition/deletion
File name: man02 Main part #: 43 Numbering: 0 to 401. Aux part #: 44 LINES
Closest points displaying
Point #: 185 # of closest points: 200 Point size: 3 Text size 15

Figure 4.15: A tool for displaying and testing the body points: (a) to locate some points on a shoulder; (b) to enumerate shoulder points, set and test the movements of these points.

As we discussed in Section 4.1 and Figure 4.1, the lower body parts are considered as one module in the implementation of modeling and simulation of these parts, we have split them into smaller functional body parts. Their tree structure is displayed in Figure 4.14b. There are ten lower body parts: the hips, thighs, shins, feet, toes, and part of the abdomen, which serves as a connection between the lower and upper body parts. This tree structure will be used to model and simulate these lower body parts and their individual movements.

Having modeled the upper and lower body parts, we incorporate the hand model into the whole body. Movements are now exerted on individual parts of the whole body depending on their motion patterns (such bending, raising, rotation), and we can simulate the movement of the human body. Several screen shots have been taken to illustrate the simulation (see Figure 4.16 and others in following chapters).

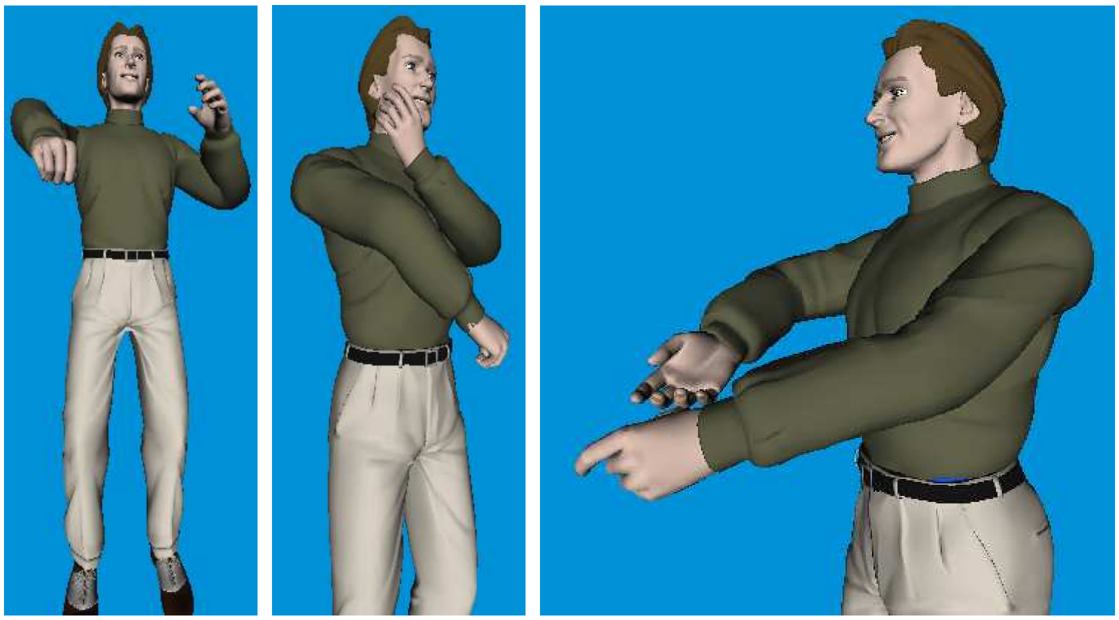


Figure 4.16: Modeling the upper and lower body parts.

4.5.2 Modeling the Head and Facial Expressions

Although there are many important biological organs that make up a head, the components of the head in our sign language interfacing system are quite simple: the mouth, eyes, hair, neck and *head*, which represents the remaining part of the human head. We use a tree structure shown in Figure 4.17 to describe the head architecture. There are six independent body part modules for modeling the head.

The eyes are modeled as balls with different colored patches on the surface, which are supposed to represent eye components such as the pupil, iris, and sclera. The simulation of the eye's movement is done by adjusting of the rotation of the eyeball at its center in different directions. The neck and hair are seen as two rigid bodies with the neck serving as the parent node in the tree structure, controlling the movement of the whole head in 3D space.

The head and mouth nodes in the head tree structure are logically separate body

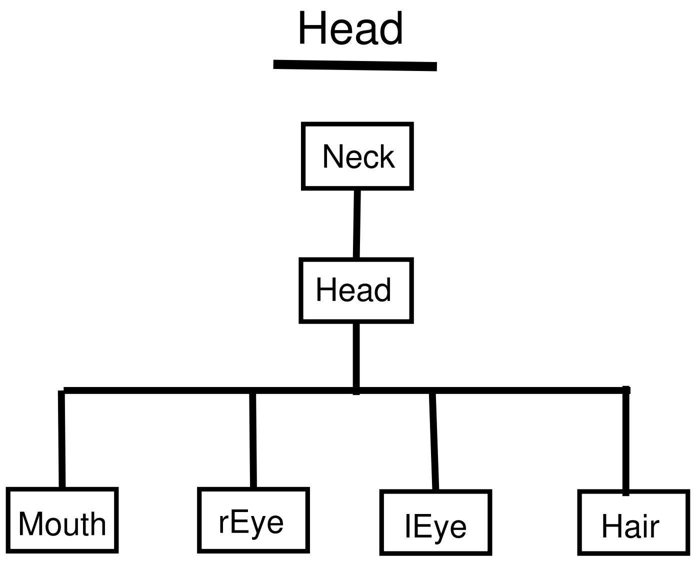


Figure 4.17: The tree structure of the head.

parts, but in this model, we consider them together with the head node responsible for the back part of the head and the mouth node for the front part. Therefore, modeling the human head and the simulation of human facial expressions in the interfacing system becomes the question of how to model the mouth node. The vertex control tool introduced previously (in Figure 4.15) can be used to locate front head points that are to be used in the synthesis of different facial expressions. Two screen shots are taken of manipulating the facial points in Figure 4.18.

According to psychological studies on human motions, there are six basic facial expressions: anger, disgust, fear, happiness, sadness, and surprise. In addition to eye movement, facial muscles move in a wide variety of different ways to produce these facial expressions. How to simulate the facial muscles in their movement is a challenging research area in computer graphics. Our solution to this problem is

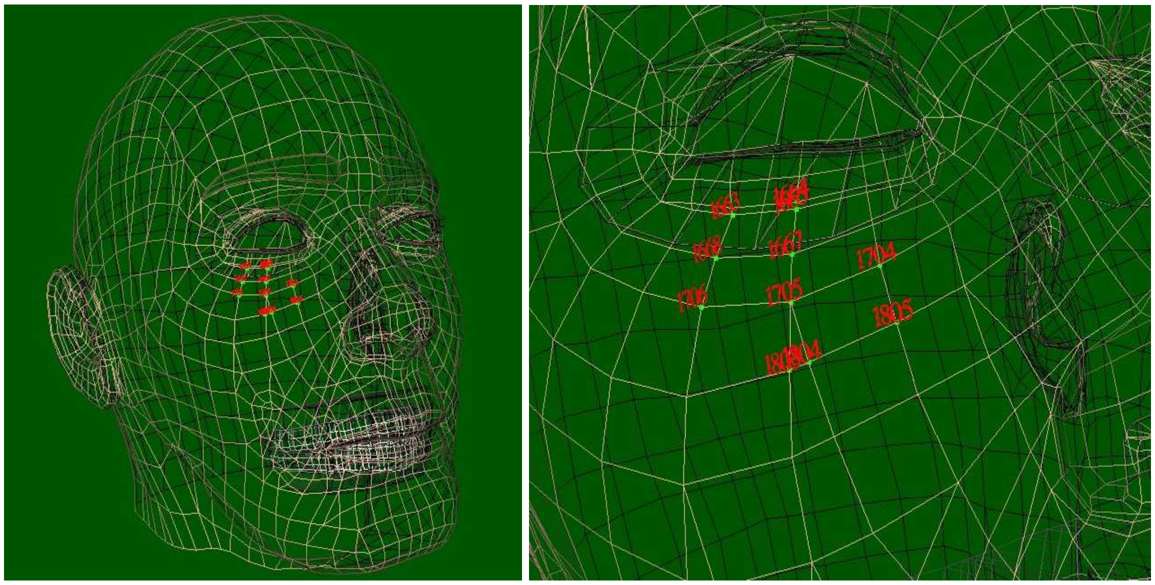
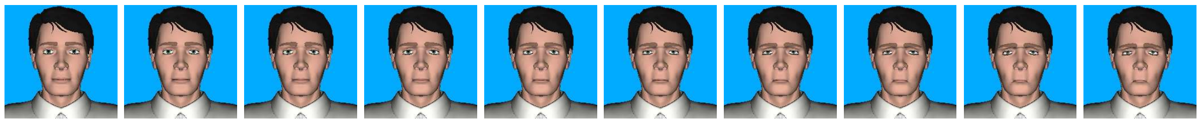


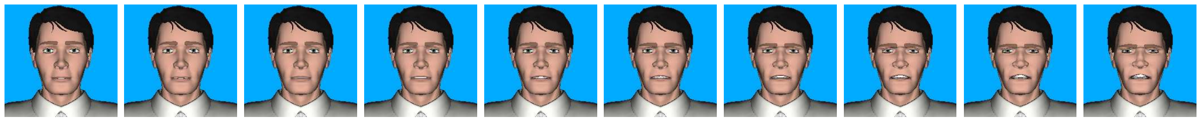
Figure 4.18: Modeling facial expressions: (a) find the related facial vertices; (b) examine the movements of the vertices.

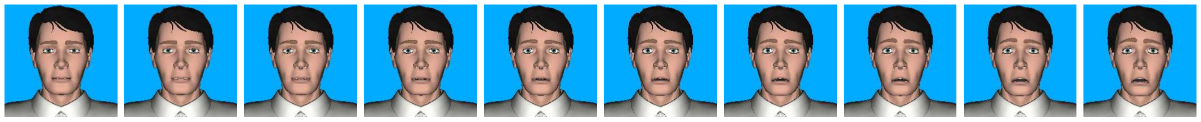
approximation.

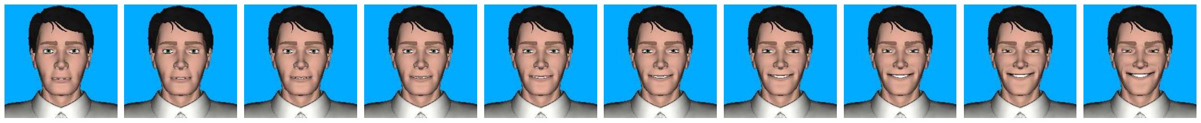
First we have obtained head data with a neutral facial expression and data for various facial expressions. Then we used the vertex control tool to identify the facial points that have changed between the neutral head and the expressive head. We defined a scale parameter to measure how great these changes were. This scale parameter is then used to adjust some facial points in a certain facial expression, and the result is head data that closely approximate that facial expression. Finally, the two sets of data (one is the neutral head) are used to model that type of facial expression with interpolation and extrapolation methods.

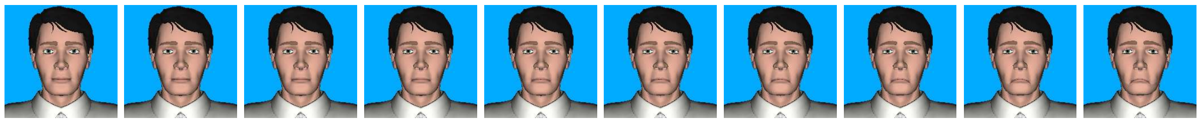
This methodology has been used to simulate the six basic facial expressions. Figure 4.19 presents some screen shots of the simulation results. We have produced 100 frames for each facial expression from a neutral expression to a facial expression at its extremity (for example, extreme happiness). Only 10 frames are displayed here for each expression.


(a) Anger.


(b) Disgust.


(c) Fear.


(d) Happiness.


(e) Sadness.

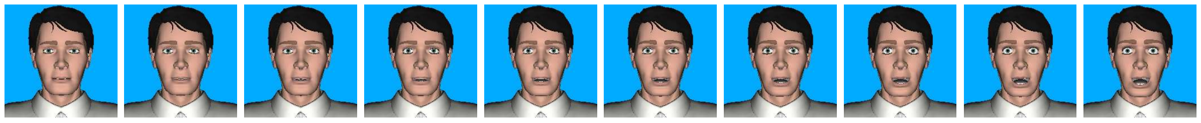

(f) Surprise.

Figure 4.19: Facial expression simulation results.

Chapter 5

Creating Virtual Gestures

We have modeled the major parts of the human body, combined them together as a virtual human body, and most importantly, simulated movement according to the motion features of the body parts. In this chapter, we will discuss how to transcribe these movements and then how to create and coordinate the movements of individual body parts for the simulation of the human gestures.

5.1 Parametric Representation of Human Gestures

A virtual human body is made of many different body parts, each of which can be implemented with an abstract representative body part, *xBody_ part*, as shown in Figure 4.3, with an extension based on its motion patterns and deformation methods. For a particular body part, its deformation algorithms and movement patterns (represented as a set of parameters) are embedded into the body object upon its substantiation. To control and simulate the movement of that body part, parametric values of the motions (translation and/or transformation) must be fed into that body part through an interface (see Figure 5.1).

As Figure 5.2 shows, when a body part receives inputs, it will interpret the inputs, based on the nature of the body part, as values for some or all motion parameters for

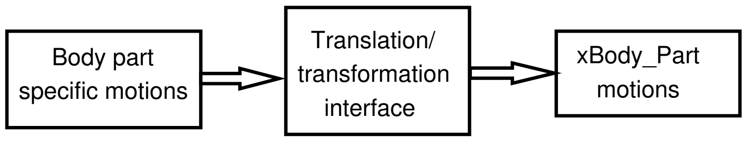

Body part specific motions
Translation/ transformation interface
xBody_Part motions

Figure 5.1: Movement control of a body part through a motion interface—the *translation/transformation interface*.

three types of rotations: abduction-adduction, twist-rotation, and flexion-extension. Then its deformation and motion mechanisms will calculate the new locations, normals, and transformation matrices for the body part's vertices according to the motion patterns and parametric values. Finally, the calculations will be fed into the body part's rendering mechanism to produce an updated image of the body part in its new position and be will output in a 4×4 transformation matrix to be used by the body part's parent node and its immediate children.

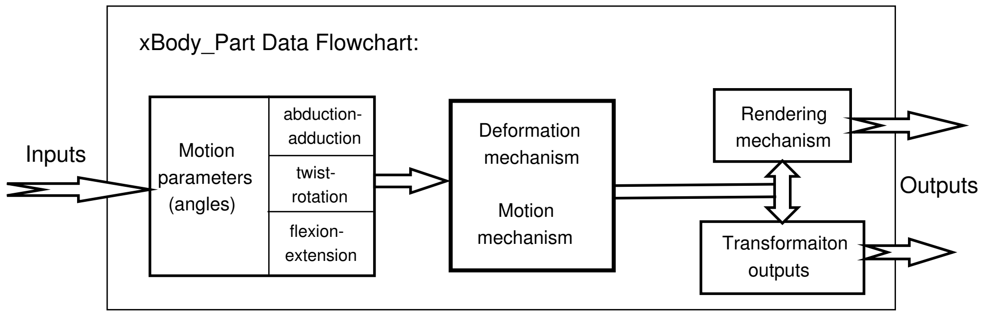


Figure 5.2: Process of how a body part module processes its input data.

Thus the movements of the virtual body can be described by and controlled with the inputs of the component body parts of the avatar, and a virtual gesture is a set of movements of the virtual body in a certain order. These inputs of the body components are a set of motion parametric variables with certain values, and therefore, a virtual gesture can be described with a cluster of sets of parametric variables. In the following sections we will illustrate the parametric representation of the gestures. But we will first give a definition of body posture and its data structure.

5.1.1 The Representation of Body Posture

Body posture means the position, pose, and bearing of the body, for example, sitting posture and erect posture. In the sign language interfacing system, we extend this definition such that *body posture* defines the positions and bearing characteristics of all the body parts including the facial expression features and hand configurations. This makes it convenient to design and implement data structures that are used to represent and process the linguistic parts of a sign language.

As illustrated in the previous section, the motion of a body part can be described with a set of parametric variables. Now we define a data structure for body posture, which has a tree structure shown in Figure 5.3.

A posture is defined with the following elements: (1) *head*, defined by the data structure *Head*; (2) ID, a unique integer number for the posture; (3) *center*, a point in 3D space (an array of three floating point numbers) for the center of the body in 3D space; (4) *orientation*, a vector for identifying the body's orientation in 3D space; (5) *rShoulder* and *lShoulder*, defined by the *Shoulder* data structure for the two shoulders (including the upper arms); (6) *rForearm* and *lForearm*, denoting the two forearms defined by the *Forearm* data structure; and (7) *rHand* and *lHand*, designating the two hands with the data structure *Hand*, which includes another two data structures: *Thumb* for the thumb and *Finger* for the fingers.

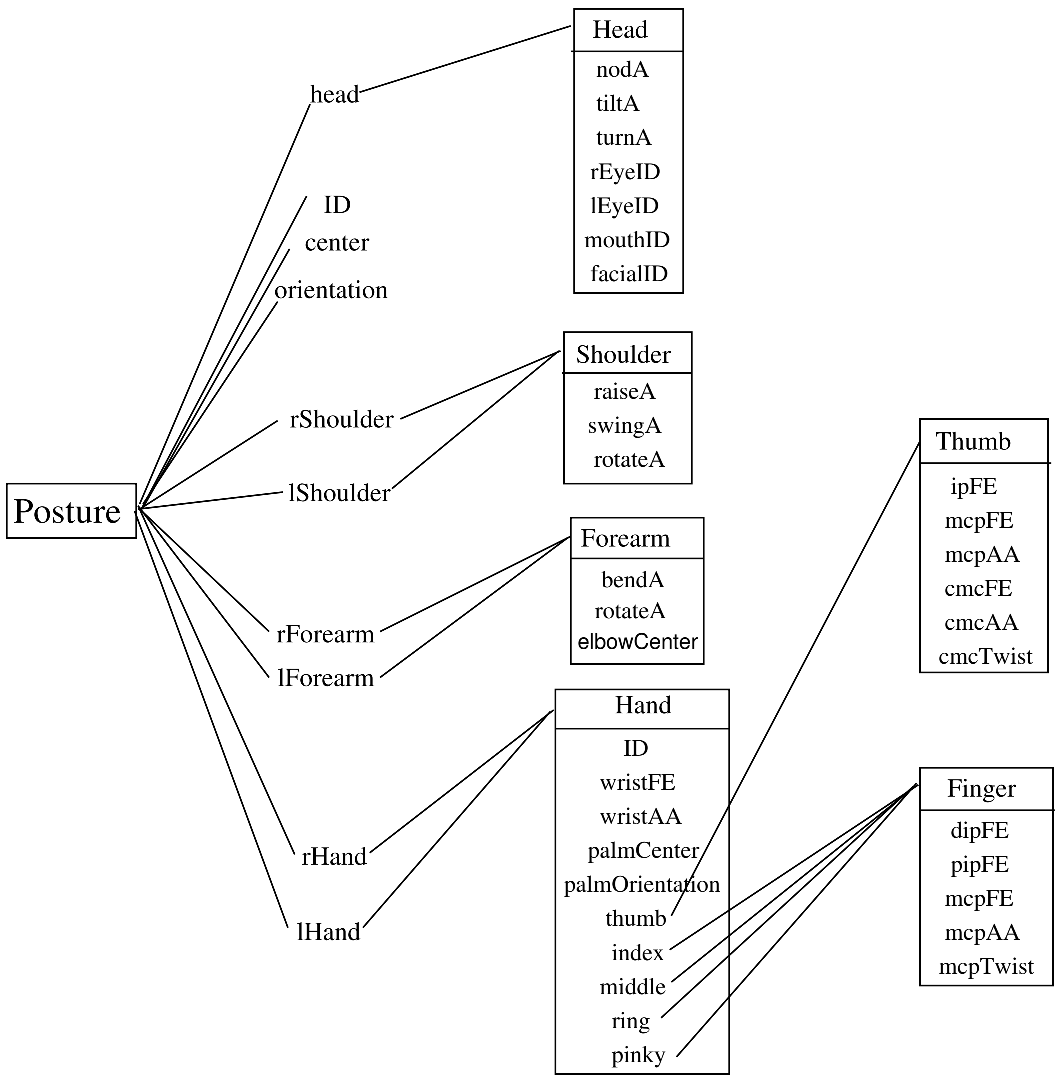


Figure 5.3: Data structure for body posture.

In the implementation of all these data structures in an object-oriented programming language (C++), we defined a *Class* for each of them, which contains a set of parametric variables depending on the body part's characteristics. The main data structures and their inclusive parametric variables are briefly explained as follows:

- *Head*: for simulation of facial expressions in addition to the control of the movements of the eyes and mouth (lips).

  - *nodA*: a floating point number for the head's nodding angle.
  - *tiltA*: a floating point number for the head's tilting angle.
  - *turnA*: a floating point number for the head's turning angle.
  - *rEyeID*: an integer that defines the right eye's orientation and other possible features (for future consideration).
  - *lEyeID*: an integer that defines the left eye's orientation and other possible features (for future consideration).
  - *mouthID*: an integer that defines the mouth openness and the lips' movements (for future consideration).
  - *facialID*: an integer representing the type of expression and the degree of the chosen facial expression (*e.g.*, a slight smile or a big smile).

- *Shoulder*: for the description and control of the shoulder's movements.
  - *raiseA*: a floating point number for the shoulder's raising angle (including the upper arm).
  - *swingA*: a floating point number for the shoulder's swinging angle.
  - *rotateA*: a floating point number for the shoulder's rotating (twisting) angle.

- *Forearm*: for the description and control of the forearm's movements and the elbow's location.
  - *bendA*: a floating point number for the forearm's bending angle.

  - *rotateA*: a floating point number for the forearm's rotating (twisting) angle.
  - *elbowCenter*: an array of three floating point numbers that gives the central location of the elbow in 3D space.

- *Thumb*: for the description and control of the movements of the three thumb parts.
  - *ipFE*: a floating point number for the flexion-extension angle of the thumb tip (the first part of the thumb).
  - *mcpFE*: a floating point number for the flexion-extension angle of the thumb's middle part.
  - *mcpAA*: a floating point number for the abduction-adduction (side-side movement) angle of the thumb's middle part.
  - *cmcFE*: a floating point number for the flexion-extension angle of the thumb's base part.
  - *cmcAA*: a floating point number for the abduction-adduction (side-side movement) angle of the thumb's base part.
  - *cmcTwist*: a floating point number for the rotation (twisting) angle of the thumb's base part.

- *Finger*: for the description and control of the movements of the three finger parts.
  - *dipFE*: a floating point number for the flexion-extension angle of the finger tip (the first part of the finger).

  - *pipFE*: a floating point number for the flexion-extension angle of the finger's middle part.
  - *mcpFE*: a floating point number for the flexion-extension angle of the finger's base part.
  - *mcpAA*: a floating point number for the abduction-adduction (side-side movement) angle of the finger's base part.
  - *mcpTwist*: a floating point number for the twisting (rotating) angle of the finger's base part.

- *Hand*: for simulation of the hand configuration in addition to the control of the movements of the hand parts.
  - *ID*: an integer to define the hand configuration.
  - *wristFE*: a floating point number for the flexion-extension (bending) angle of the wrist.
  - *wristAA*: a floating point number for the abduction-adduction (side-to-side movement) angle of the wrist.
  - *palmCenter*: an array of three floating point numbers that gives the central location of the palm in 3D space.
  - *palmOrientation*: a vector (an array of three floating point numbers) that defines the orientation of the hand (the palm).
  - *thumb*: for the description and control of the thumb movements.
  - *index, middle, ring* and *pinky*: for the description and control of the movements of the fingers of the hand.

The *elbowCenter* and *palmCenter* data structures record the positions of the hand and elbow and play an important role in classifying the virtual gestures and searching a sign language for linguistic parts. As a new posture is constructed, the positions for the elbow and the hand are automatically calculated by the system and become a part of the parametric representation of the posture.

The *nodA, tiltA, turnA, rEyeID, rEyeID*, and *mouthID* are reserved for future use. In the current version of the interfacing system, we have modeled and simulated the six basic facial expressions, but have not combined them into the system.

5.1.2 The Representation of Body Gestures

People make gestures by starting with a posture and ending with another posture, assuming a series of varying postures in between. In a similar way, a virtual gesture can be described as an ordered set of postures of a virtual human body. A timing factor is thus introduced to describe the order of the postures: posture p_i occurs at time t_i where p_i is a parametric representation of the posture at time t_i and has a data structure we defined in the previous section. Thus we use a list to define the gesture, $vg = [(p_0, t_0)$, (p_1, t_1), ..., $(p_n, t_n)]$ in which p_0 is the starting posture at time t_0 and p_n is the ending posture at time t_n. Figure 5.4 shows the data structure of a virtual gesture.

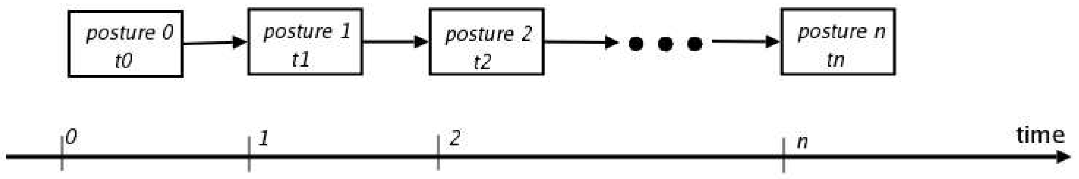


Figure 5.4: The formation of a virtual gesture.

There is a big difference between a human's gesture in real life and the gesture defined above: the former is a *continuous* process, which means an infinite number of postures in gesturing session, while the later is only a limited number of postures in a posture list. We solve this problem in two steps. First, for a virtual gesture, a group of distinctive postures is selected that reflect characteristics of the gesture; next temporary postures between two adjacent postures are interpolated during the output process of the virtual gesture based on the rendering speed (frames per second) and time difference of the two adjacent postures.

The interpolation algorithm will be discussed later in this chapter.

5.2 Construction and Management of Body Postures

As illustrated in the last section, body posture in the sign language interfacing system is interpreted as a set of parametric variables that describes the distinctive features of a body's bearing. According to the characteristics of the body parts and their functionalities in a signing process, we classify the parametric representations of the body parts into three separate groups: hand configuration, upper limb positioning, and NMS (nonmanual signal) as shown in Figure 5.5.

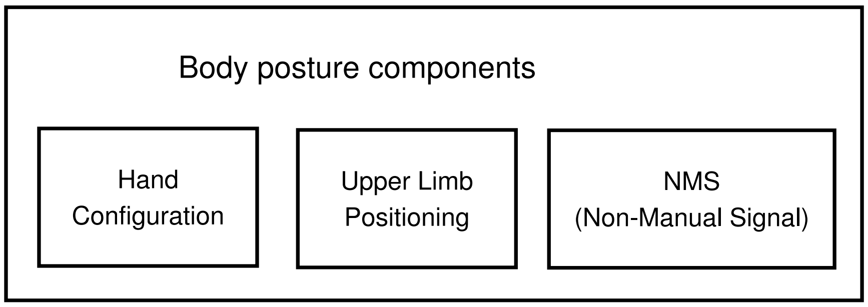
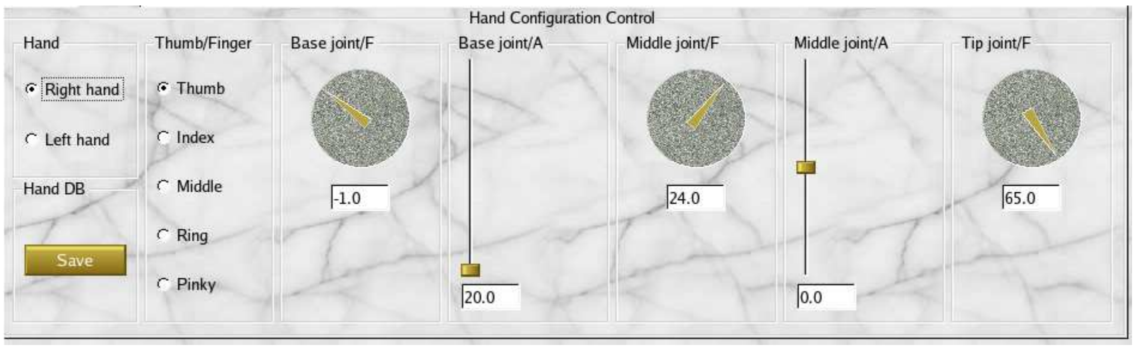


Figure 5.5: Body posture components.

In this section we will discuss the construction and management of the virtual postures in the interfacing system. The basic idea is to create an efficient graphical user interface for providing and adjusting parametric values for the posture's representative parametric variables. The management of body postures such as storing, editing, and retrieving of the postures are handled by a posture database. We will first talk about the construction and management of hand configurations and then that of upper limb positioning. We leave the creation of nonmanual signals for the next chapter.

5.2.1 The Hand Configurations

A *Hand Configuration Control* panel has been built and embedded into the sign language interfacing system (see Figure 5.6) for the creation and editing of hand shapes. Graphical widgets are used to provide and adjust values (degrees of rotation angles) of the parametric variables for a certain hand configuration.



Hand: [✓ Right hand] Left hand
Thumb/Finger: [✓ Thumb] Index Middle Ring Pinky
Base joint/F: -1.0
Base joint/A: 20.0
Middle joint/F: 24.0
Middle joint/A: 0.0
Tip joint/F: 65.0

Figure 5.6: Hand configuration control panel.

The radio button group (with the widget titled *Thumb/Finger*) consists of buttons named *Thumb, Index, Middle, Ring* and *Pinky* and is used for the selection of one of the four fingers or the thumb. The three iconic dials are used to control the bending (flexion-extension) movements at the three joints of the chosen finger or thumb. The vertical slide-bar with the name *Base joint/A* is used for both the finger and thumb's side-to-side (abduction-adduction) movements at their base joints; the other slide-bar, named *Middle joint/A*, is exclusively for the side-to-side movement of the thumb at its middle joint (when one of the four fingers is chosen with the radio buttons, this slide-bar will become grayed, indicating ineffective). If accurate control of the hand joints' movements is needed, users can input values for the joints' rotation angles in the spaces below the dials and slide-bars.

A hand configuration database is used to assist in the creation, editing, and management of hand configurations as shown in Figure 5.7, which serves as a flow chart of the hand configuration sub-system. At first we have constructed several dozen hand shapes, put them into the hand configuration database, and embedded them into the sign language interfacing system (see Figure 3.2).

To create a new hand shape, we first search for a basic hand configuration in the database that has a similar pattern. If we cannot find one, we use a default hand shape with a neutral position. Then we use the hand configuration control panel to finetune the angles of the hand's joints. Finally, we use *Save* button (see Figure 5.6) to save the newly created hand configuration in the database.

When we want to apply a hand configuration to the left hand or right hand of the virtual body, we first select which hand with the radio button group with the title *Hand* on the panel (see Figure 5.6). If the hand configuration is in the hand configuration database, use it; if not, first search for a close one in the database and then, adjust its rotation angles to create till the correct hand shape.

Every created hand configuration begins with a neutral hand (palm) orientation, which will later be changed by adjusting the movements at the wrist's joint. The

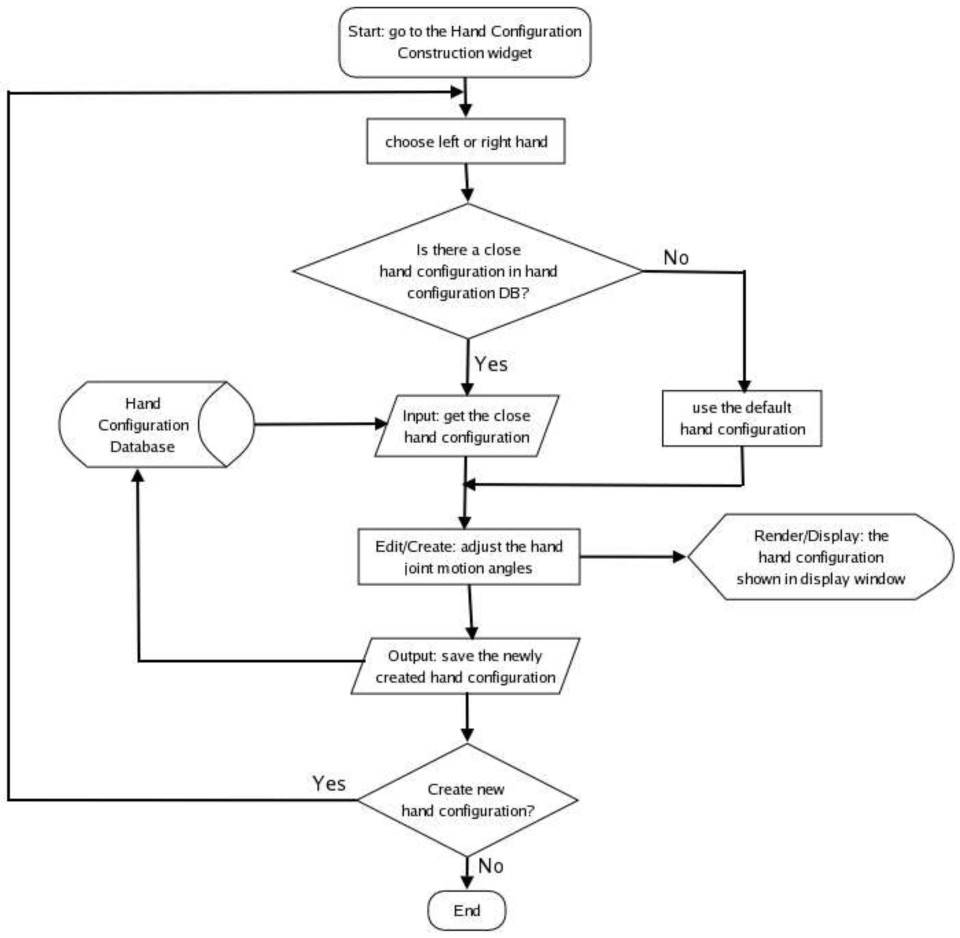

Start: go to the Hand Configuration Construction widget
choose left or right hand
Is there a close hand configuration in hand configuration DB?
No
Yes
Hand Configuration Database
Input: get the close hand configuration
use the default hand configuration
Edit/Create: adjust the hand joint motion angles
Render/Display: the hand configuration shown in display window
Output: save the newly created hand configuration
Yes
Create new hand configuration?
No
End

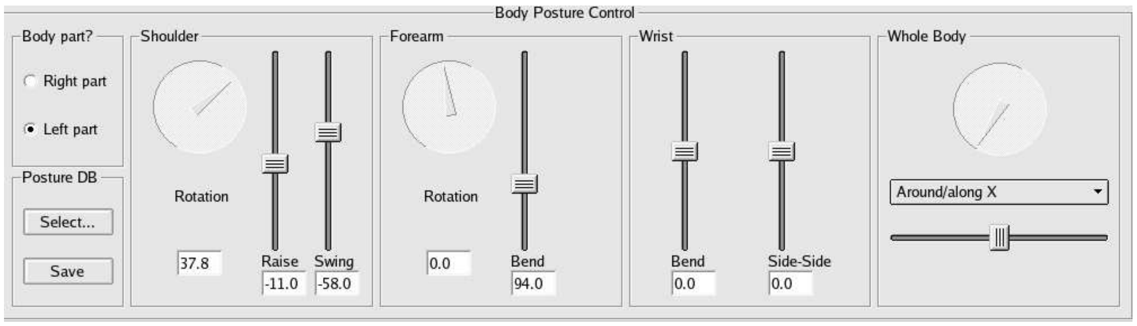
Figure 5.7: Flow chart for creating and editing hand configurations.

right and left hand parts of the virtual body in the sign language interfacing system are symmetric to each other, therefore the hand configuration database needs to save only one set of data for each hand shape. When a hand configuration is applied to a chosen hand (right or left), the system will automatically adjust the hand data accordingly.

The process and results of hand configuration creation and application (from the database) will be displayed in display windows (as shown in Figure 3.2).

5.2.2 The Upper Body Postures

The upper body parts of the virtual figure in the interfacing system include the shoulder (together with the upper arm), forearm, and wrist joint, which is responsible for the hand (palm) orientation. A body posture control panel has been constructed and incorporated into the sign language interfacing system for providing inputs for upper body part joints as shown in Figure 5.8, using the data structure defined in the last section.



Body part?: Right part [✓ Left part]

Shoulder: Rotation 37.8; Raise -11.0; Swing -58.0

Forearm: Rotation 0.0; Bend 94.0

Wrist: Bend 0.0; Side-Side 0.0

Whole Body: Around/along X

Figure 5.8: Upper body posture control panel.

There are three widget groups on the panel for controlling the upper body movements: (1) *Shoulder*: the shoulder and upper arm's twist (rotation) is adjusted with a iconic dial, and the raising and swing movements are adjusted two vertical slide-bars; (2) *Forearm*: the forearm's rotation (twist) is adjusted with a dial widget, and its bending movement is controlled with a vertical slide-bar; and (3) *Wrist*: the wrist's bending and side-to-side movements are adjusted with two vertical slide-bars. Input spaces (below these widget icons) are available for accurately positioning the upper body parts. All these widget groups are under the control of another widget group on the same panel entitled *Body part*, which indicates whether the left body part or the right one will get inputs from the three widget groups. There is another widget group that control the whole body's movements: displacements (translations) along and rotations around three perpendicular axes of the coordinate system for the whole body.

Figure 5.9 is a flow chart of the sub-system *body posture creation and manage-*

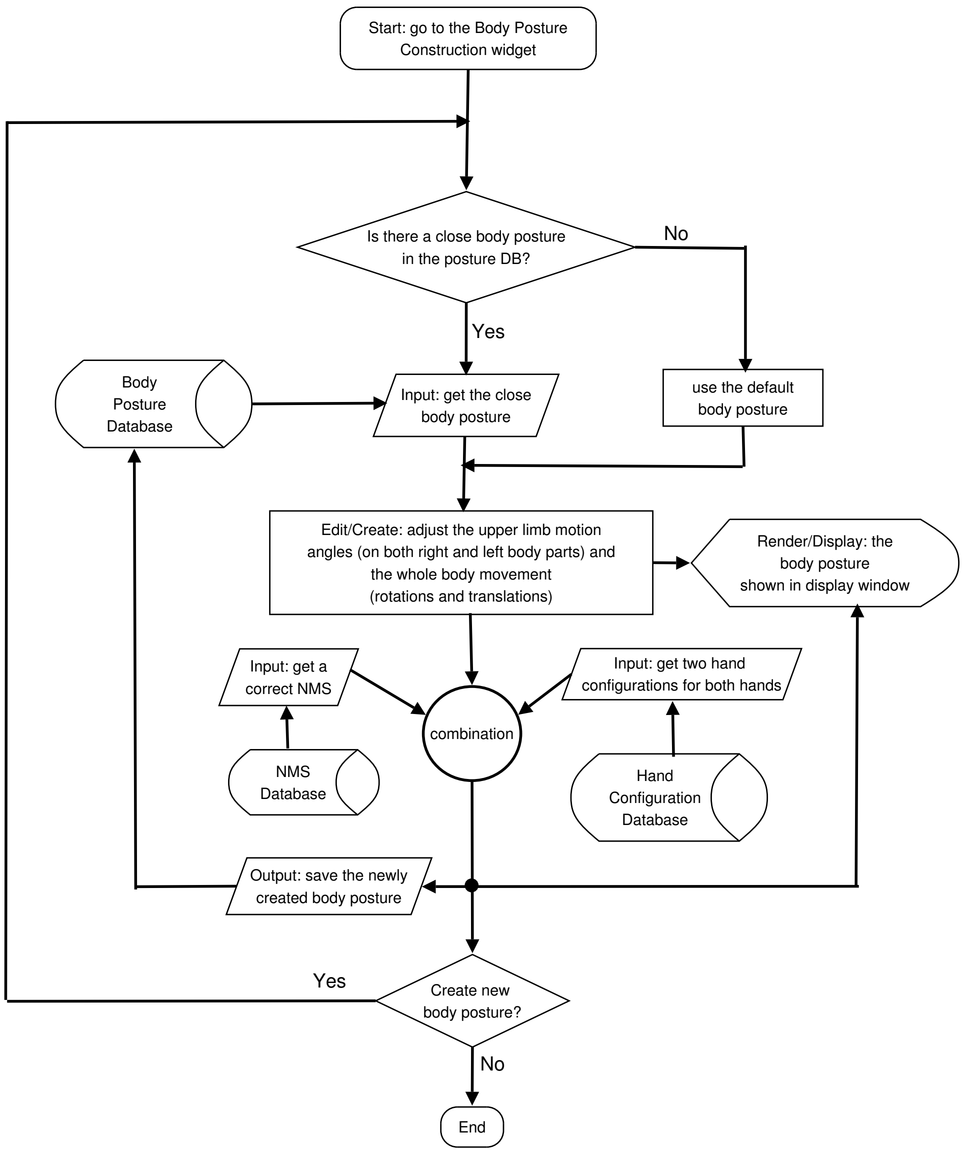

Start: go to the Body Posture Construction widget
Is there a close body posture in the posture DB?
No
Yes
Body Posture Database
Input: get the close body posture
use the default body posture
Edit/Create: adjust the upper limb motion angles (on both right and left body parts) and the whole body movement (rotations and translations)
Render/Display: the body posture shown in display window
Input: get a correct NMS
combination
Input: get two hand configurations for both hands
NMS Database
Hand Configuration Database
Output: save the newly created body posture
Yes
Create new body posture?
No
End

Figure 5.9: Flow chart for creating and editing upper body posture.

ment. A body posture database is used together with the NMS and hand configuration databases for storing, editing, and retrieving body postures. Some basic body postures have been constructed and stored in the posture database.

When a new body posture is created, all the parametric values corresponding to the posture's data structure (as defined in the last section) can be stored in the body posture database with a push of the *Save* button on the panel (as shown in Figure 5.8). Of all these parametric values, two groups stand out for their importance in categorizing and designing virtual gestures: the location of the center of each hand palm and the location of the center of each elbow in the coordinate system for the whole virtual body. These locations are automatically calculated following the tree structure of the whole body as the inputs are fed into the body posture control panel.

If we want to change the avatar's posture, the first step is to try to find a close body posture in the posture database (with the use of the *Select* widget on the panel) or use a default body posture when no close posture is found in the database. Then use the body posture control panel to adjust the positions of the upper body parts. The NMS database and hand configuration database are searched for the correct facial expression and hand shape. Throughout the construction process, the current posture is displayed in the display windows of the interfacing system. There are virtual cameras to choose from and their foci can be adjusted for clear displaying from various viewing directions (see Figure 3.2).

5.3 Creation and Management of Virtual Gestures

As described in the first section of this chapter and in Figure 5.4, a virtual gesture, vg, is an ordered list of postures; that is, $vg = [(p_0, t_0), (p_1, t_1), ..., (p_n, t_n)]$ with p_i being the ith posture in vg at the time t_i. The postures p_0, p_1, ..., p_n constitute a complete set of postures for a given gesture and are the most representative and characteristic postures for that the gesture, which describe fully the gesture process. Once a posture list vg is extracted for a certain gesture, intermediate and temporary postures can be interpolated between any two adjacent postures in the list for displaying and output. The problem of creating a virtual gesture becomes how to construct such a posture list given the gesture. We will first introduce a prototype to explain the design architecture for creating and editing virtual gestures, then present our implementation with the gesture creation and editing control panel, and finally use an example to illustrate how to construct and edit virtual gestures.

5.3.1 Design Architecture for Creating and Editing Virtual Gestures

We use Figure 5.10 to illustrate the design architecture for creating and editing

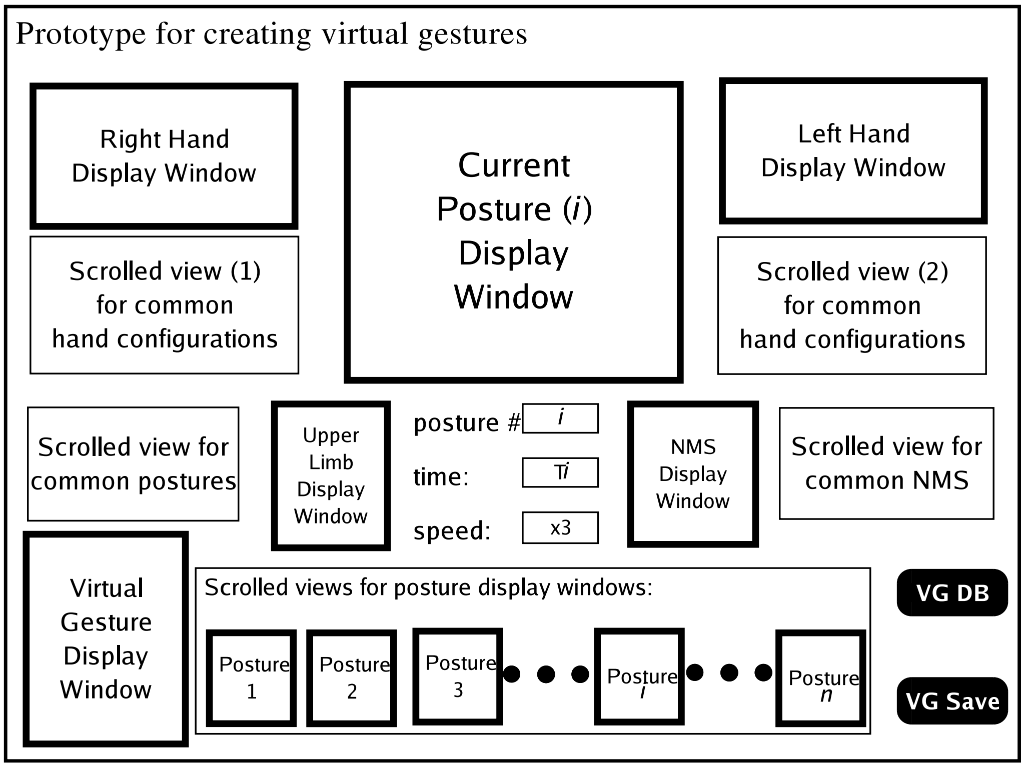

Prototype for creating virtual gestures
Right Hand Display Window
Current Posture (i) Display Window
Left Hand Display Window
Scrolled view (1) for common hand configurations
Scrolled view (2) for common hand configurations
Scrolled view for common postures
Upper Limb Display Window
posture # i
time: Ti
speed: x3
NMS Display Window
Scrolled view for common NMS
Virtual Gesture Display Window
Scrolled views for posture display windows:
Posture 1
Posture 2
Posture 3
Posture i
Posture n
VG DB
VG Save

Figure 5.10: A prototype for creating virtual gestures.

virtual gestures. In the last section we illustrated the creation and management of hand configurations and upper body postures with their corresponding databases. The various facial expressions or nonmanual signals (NMS), in a similar way, are stored in and retrieved from an NMS database (which will be discussed in next chapter). With these databases and control panels for instant construction of the posture components for special cases, users can quickly build a virtual gesture if assisted with an efficient graphical user interface wrapper that combines all the operations of the gesture creation process. In this sense, Figure 5.10 serves as a prototype for such a wrapper.

In this prototype diagram, the boxes with bold edges represent display windows for displaying the temporary and overall results in the gesture construction process. In the upper part of it (compare with Figure 3.2) there are right and left hand display window on either side with a main display window for displaying the current posture of the whole body. In the lower part, there are NMS and upper body posture display windows, the virtual gesture display window (for displaying the gesture animation process), and posture display windows (for displaying all the postures of the gesture).

The other boxes are used for the interactions with the databases and operations on gesture controls (such as setting time and speed). Figure 5.11 is an illustrative diagram for the wrapper prototype, and we will use it to explain how to construct a virtual gesture. When users are going to construct a virtual gesture, they have an imaginary gesture in mind. What they need to do first is to select from repertoires of upper body postures, hand configurations, and NMSs—the representative components for the gesture. If some components (upper body postures, hand shapes, or NMS) do not exist in the databases, users can use the corresponding control panels to create and save them to the databases. A typical case is that there are some components that are close the correct ones, and users can select these and use the control panels to finetune them. In the following discussion we assume that these components exist in the databases.

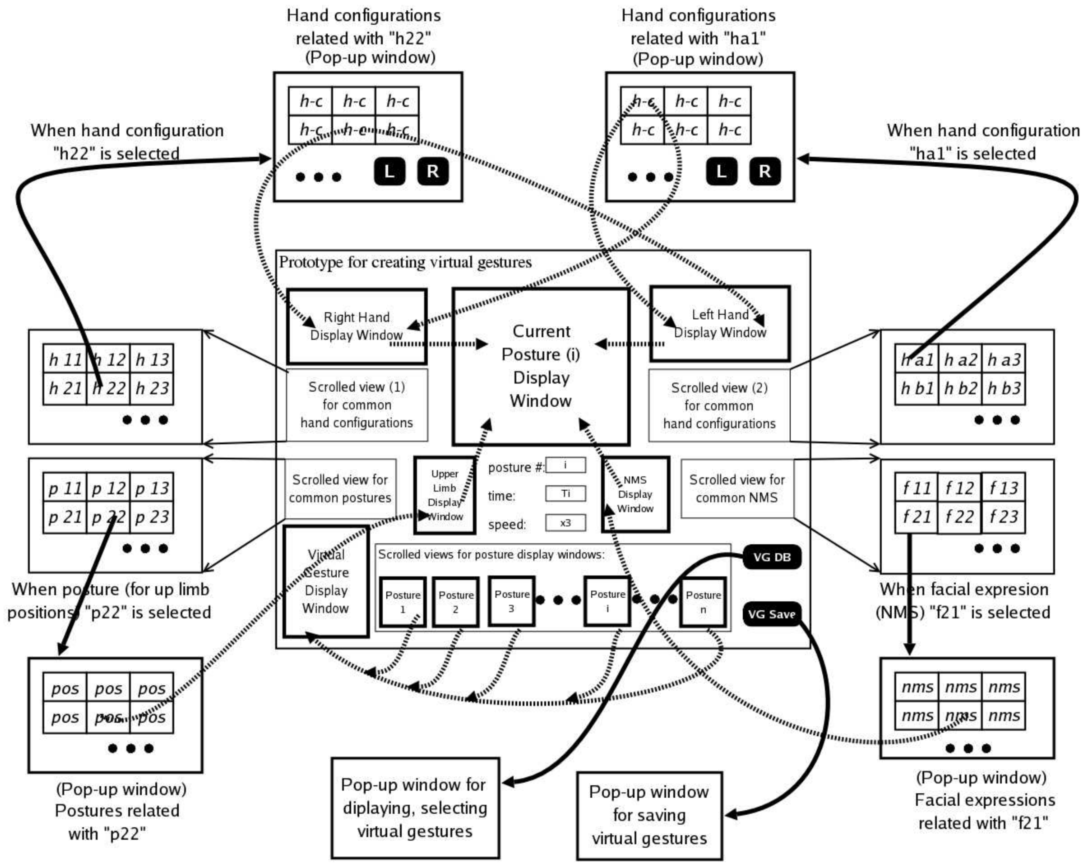

Hand configurations related with "h22" (Pop-up window)
Hand configurations related with "ha1" (Pop-up window)
When hand configuration "h22" is selected
When hand configuration "ha1" is selected
h-c
L
R
Prototype for creating virtual gestures
Right Hand Display Window
Current Posture (i) Display Window
Left Hand Display Window
Scrolled view (1) for common hand configurations
Scrolled view (2) for common hand configurations
h 11 h 12 h 13 h 21 h 22 h 23
h a1 h a2 h a3 h b1 h b2 h b3
Scrolled view for common postures
Upper Limb Display Window
posture #: i
time: Ti
speed: x3
NMS Display Window
Scrolled view for common NMS
p 11 p 12 p 13 p 21 p 22 p 23
f 11 f 12 f 13 f 21 f 22 f 23
Virtual Gesture Display Window
Scrolled views for posture display windows:
Posture 1
Posture 2
Posture 3
Posture i
Posture n
VG DB
VG Save
When posture (for up limb positions) "p22" is selected
When facial expresion (NMS) "f21" is selected
pos
nms
(Pop-up window) Postures related with "p22"
Pop-up window for diplaying, selecting virtual gestures
Pop-up window for saving virtual gestures
(Pop-up window) Facial expressions related with "f21"

Figure 5.11: A diagram illustrating how to create virtual gestures.

In Figure 5.11, a dotted curve means the transfer of image results. The right/left hand configurations, NMS component, and upper postures can be displayed in their display windows *and* the main display window. A solid bold curve represents the path of the result when a user clicks on a widget (an image icon or a button) on the panel. A pair of solid thin (arrowed) lines shows an expanded view of some of the boxes (*e.g.*, the box *Scrolled view for common NMS* contains the $f11$, ..., $f23$ which are icons for various facial expressions). For example, as a user clicks the push button *VG DB*, a pop-up window will appear containing many virtual gestures for the user to choose from.

When a user is about to insert an ith posture into the virtual gesture, she/he first clicks on one posture icon (in Figure 5.11, icon $p22$) in the box *Scrolled view for common postures* containing many common upper posture icons ($p11$, $p12$, ..., $p23$..., each of which is a posture image), and a pop-up window will appear containing postures close to (or related to) the posture $p22$, as the solid bold line indicates. The user then chooses one posture from this pop-up posture window, and the selected posture will replace the old posture of the virtual body and be displayed in both the *Upper Limb Display Window* (for close-up displaying) and the *Current Posture (i) Display Window* (for display of the whole body).

The process is repeated for selecting an NMS from the NMS database and for choosing right/left hand shapes from the hand configuration database. As for the latter case, *Scrolled view (1) for common hand configurations* and *Scrolled view (2) for common hand configurations* are used for both hands. For instance, the two pop-up hand configuration windows on the top of Figure 5.11, which display the related hand configurations related to hand shape $h22$ (on the left side) and hand shape $ha1$ (on the right side), can be used for right/left hand configuration selection with the push buttons labeled R or L.

When this process is done, the posture i (with an upper posture, an NMS, and right/left hand configurations) has been inserted into the virtual gesture. *Virtual Gesture Display Window* will automatically show the gesture animation session based on the *speed* setting. On the right of the virtual gesture display window are display windows for all the postures of the current gesture. A user can click any of them to edit the gesture with the posture control panels and change the time setting. The newly created virtual gesture can be saved to the gesture database with push button *VG Save*.

5.3.2 A Virtual Gesture Creation/Editing Panel

With the use of the gesture creation prototype shown in Figure 5.11 as a guide, we have implemented a virtual gesture creation and editing interface (see Figure 5.12) and incorporated it into the sign language interfacing system. Users use this interface, together with the posture control panels described in last section, to create, edit, store, and retrieve any virtual gestures.

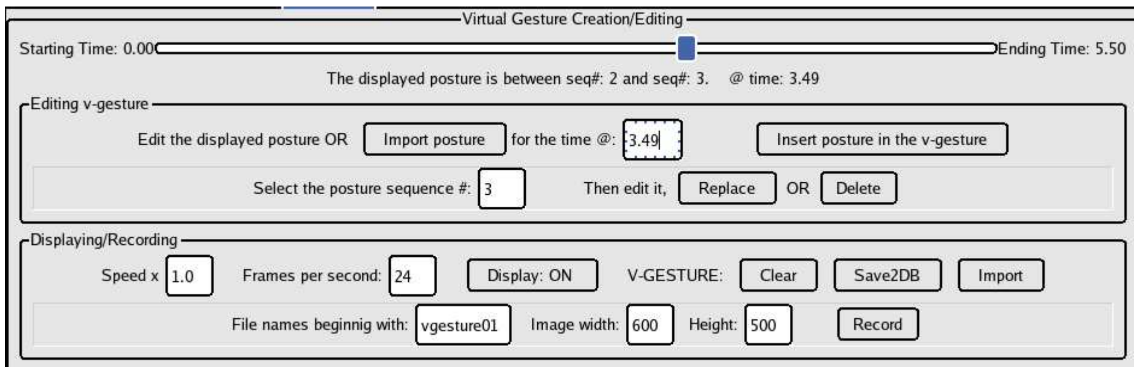


Figure 5.12: Virtual gesture creation/editing panel.

At the top of the gesture interface, there is a posture sequence sliding bar together with a gesture information line below it. Users use the sliding button on the bar to display in the display windows (on the upper part of the interfacing system, see Figure 3.2) the postures of the current gesture, either those representative postures (p_0, p_1, ..., in the vg) or any temporary interpolated ones between any two adjacent postures in the vg. The middle part of the interface is used for posture editing functions: importing (from the posture database), editing, replacing, deleting, and inserting (into the current gesture). The bottom of the interface is reserved for displaying, saving, and recording the current gesture.

At the beginning, the gesture is made up of any two postures distributed at time 0 and 2 (seconds). Users can replace these postures with postures from the posture database with graphical widgets such as the push buttons *Import posture, Replace, Insert posture in the v-gesture* and the input space *Select the posture sequence #* to type in which posture needs to be edited or replaced. The timing factors for new postures are input in space entitled *for the time @.* The current posture can also be deleted with the *Delete* button.

The gesture session can be displayed dynamically depending on the status of the *Display* switch button, which appears in only one mode: *Display: ON* or *Display: OFF.* The gesture display sliding bar and the sliding button on it can be used for accurate control of the gesture postures.

5.3.3 An Example of Creating and Editing a Gesture

Suppose we are about to create a gesture that has the four characteristic postures shown in Figure 5.13. These postures (from left to right in this figure) will appear in the gesture at time (in seconds) 0, 1.5, 2.8, and 4.2. We construct these postures (or load them from the posture database) and insert them in the gesture and delete a default posture at time 2.0. Now we turn on the *Display* switch button, and the display windows will display the gesture animation process, which lasts 4.2 seconds with the default speed of 24 frames per second. In this case, the interfacing system will have automatically interpolated about 97 intermediate postures for this gesture ($24 \times 4.2 - 4 \approx 97$).

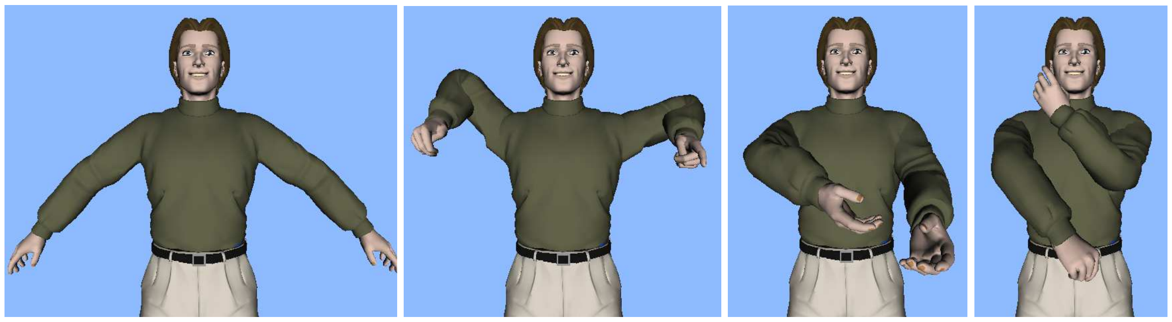


Figure 5.13: A virtual gesture: its four posture component.

In watching the gesture process, we noticed that something seemed wrong during past of the gesture session. We turned off the display and used the gesture display sliding bar and sliding button to examine some postures of the gesture. We discovered that at 3.12 seconds we had in the interpolated postures the unnatural one shown shown in Figure 5.14a (along with other unnatural postures before and after this

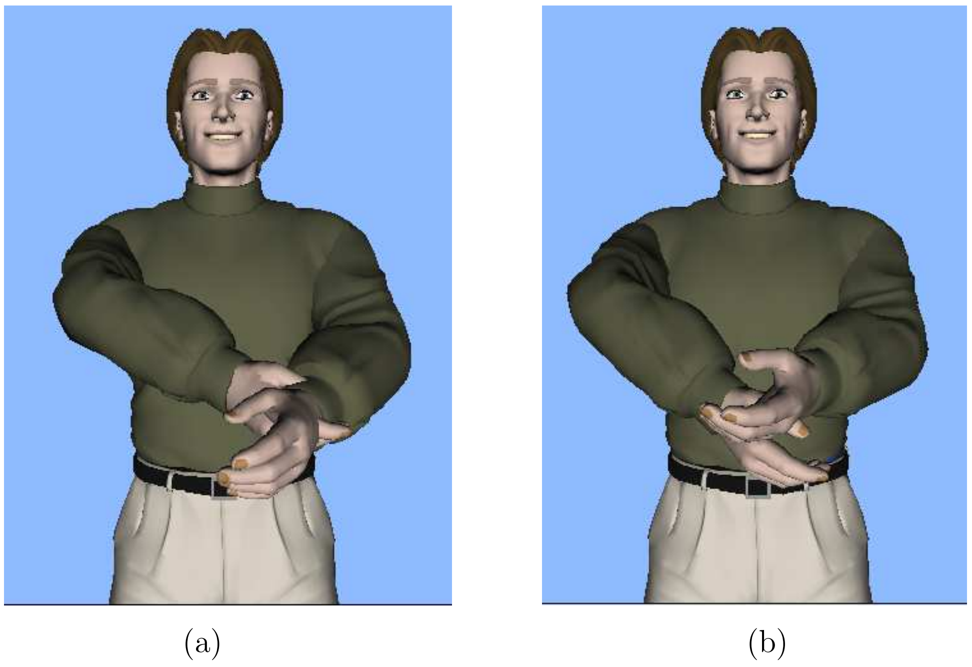


Figure 5.14: How to create a natural gesture: (a) unnatural temporary postures interpolated during virtual gesture rendering; (b) choose a representative posture of these unnatural postures, edit it, and insert it to the gesture as a component posture for that gesture.

posture).

We immediately went to the posture control panels and edited this posture by adjusting the left and right shoulders' (and forearms') positions with the widgets on the *Body Posture Control* panel (as described in the last section). These changes resulted in a new and natural posture shown in Figure 5.14b. Now, we simply clicked on the button *Insert posture in the v-gesture*, and this new posture (once a temporary and interpolated posture) became a member (at 3.12 seconds) of the gesture.

When we turned on the gesture display button, the interfacing system automatically interpolated new intermediate postures for the edited gesture. The result is the same number of postures for the gesture with some of the postures updated. Figure 5.15 shows some of these posture. The gesture can be saved into the gesture database with the button *Save2DB* and recorded in image files with the button *Record*.

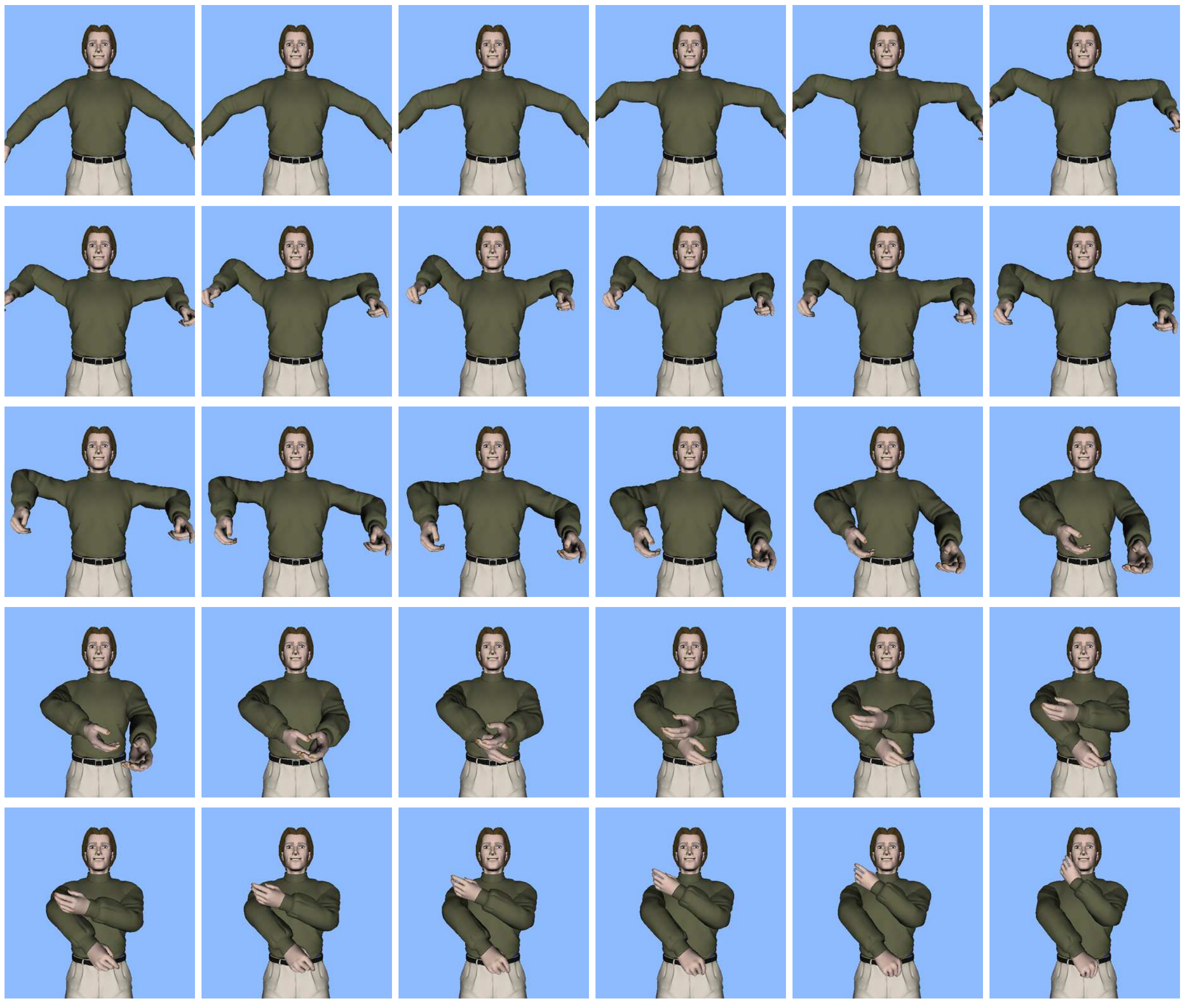


Figure 5.15: An example of a virtual gesture sequence.

Chapter 6

Conveying Linguistic Meaning

We have presented modeling of the human body, simulation of the movements of the body parts, representation of a virtual human body as a set of parametric variables, and definition of data structures to describe and implement virtual gestures of a virtual figure. Several graphical user interfaces have been introduced to create and manage virtual postures and gestures. Now we are going to apply these results in designing and implementing a virtual signing system—the proposed *sign language interfacing system.*

A signer of a particular sign language makes gestures according to the grammar of that sign language; an avatar can also imitate this process by following commands on the movements of the virtual body parts if these movements are designed to abide by grammatical rules of that sign language. Thus a virtual gesture session, *virtual signing*, acquires a meaning, and the virtual body makes virtual signs.

In this chapter, we will introduce a platform that users of sign languages can use to build basic linguistic parts (such as "phonemes" and "morphemes") of sign languages, create sign language vocabularies, and even "write" in a sign language. America Sign Language (ASL) is used as an example for the introduction.

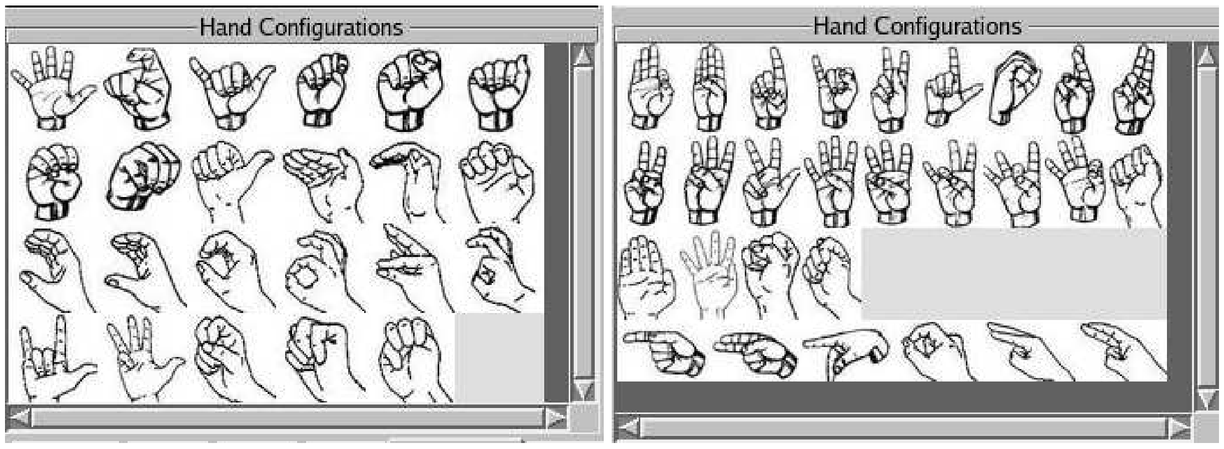
6.1 Constructing Basic Linguistic Parts

The concept of "articulatory bundle" [60], which describes hand posture with hand configuration, point of contact, facing, and orientation, provides good guidelines for designing virtual signing units. However, it is more effective to use graphical designs and implementations when dealing with the five basic linguistic parameters of a sign language: location, handshape, orientation, movement, and nonmanual signals (NMS) [94]. We have also considered "local movement," a special case of the movement parameter. The Movement-Hold model [60] is embedded in the graphical implementation of phonological and morphological parts.

The five basic linguistic parts (parameters) can be simulated with the movements of some of the virtual body parts. They are special body gestures and therefore can be represented with a list of postures combined with timing factors, as we described in the previous chapter: $lp = [(p_0, t_0),\ (p_1, t_1),\ ...,\ (p_n, t_n)]$ (lp represents any of the basic linguistic parts). We will use the list lp to describe the design and the basic parameters. The question now is how to quickly construct for lp the p_i's and combine these p_i's with their t_i's. Our solution is to design and implement efficient GUI wrapper of all the operations needed for the creation and editing of the basic linguistic parts.

6.1.1 Hand Shapes and Orientations

The hand shape is the most important phonological part of ASL and other sign languages; thus we have constructed some of the most frequently used hand shapes (as shown in Figure 6.1) and embedded them in the sign language interfacing system. These hand shapes are extracted as hand icons and put in two scrolled view areas in the upper part of the system interface (see Figure 3.2) so that users can reach these common hand shapes when they are in any of the control panels of the system.



(a) (b)

Figure 6.1: Commonly used hand shapes in sign languages.

The hand shapes can be applied to both the left and right hands of the avatar in the system: users first decide on the correct hand by using the radio button (see Figure 5.6) and then clicking on the hand icon of the hand shape they want to apply to the avatar. The virtual hand shape corresponding to the hand icon will be applied to the chosen hand of the virtual body; the two display windows (the main window and one of the two smaller display windows—depending on which hand) will show the result promptly.

The two scrolled view areas for hand shapes are open for the addition of new hand shapes. This is done by: (1) building a new hand shape with the hand configuration control panel, (2) drawing a hand image (icon) for the hand shape (or simply screen-catching the hand shape from the hand display window) and saving it to an image file, (3) clicking on the *Save* button of the panel's widget entitled *Hand DB* (see Figure 5.6), (4) inputting in a pop-up window box the information related to the newly created hand shape (such as the hand icon's location in the scrolled view boxes and the hand image/icon file name), and (5) rebooting the system.

For some hand shapes that have many variations and/or other hand shapes related to them, there will not be enough space in the scrolled view areas to display them. One solution would be that clicks on these hand icons in the scrolled area will activate a pop-up window box, in which there are hand icons for those related hand shapes and variations. Clicking on one of these icons will result in the application of its corresponding hand shape in the system avatar.

The hand (palm) orientation is dependent on the movements of the other body parts (such as forearms) and is relatively independent of hand shapes. The movements at the wrist joint also affect the hand orientation. Thus we use a neutral orientation as a default for all the hand shapes before their application to the avatar. When being applied to the virtual body, a hand (shape) immediately takes on the orientation defined by the other body parts. Furthermore, a hand orientation can be changed with the *Bend* and *Side-Side* slide-bars in the *Wrist* widget group in the *Upper Body Posture Control Panel* (see Figure 5.8).

6.1.2 Gesture Space and Locations

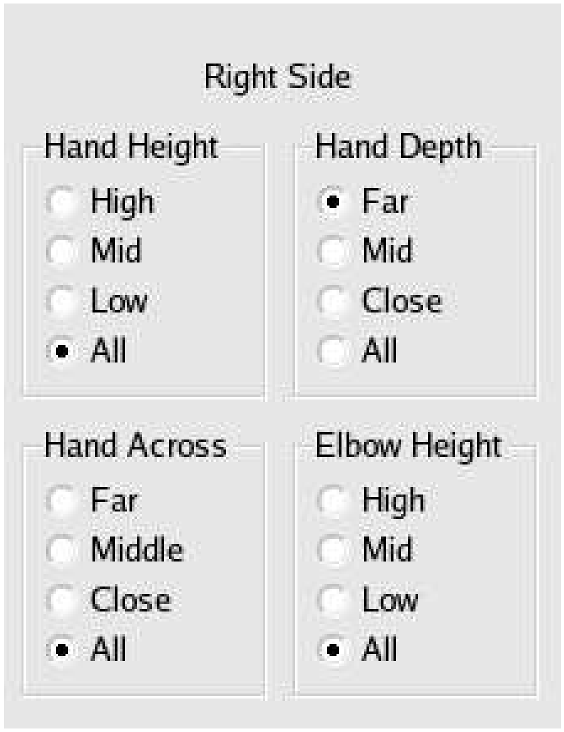
The gesture space is the space domain of the hand motions when people make gestures. According to McNeill's studies on human gestures, the gesture space is "a shallow disk in front of the speaker" [66]. McNeill divided the gesture space into different sectors and found that different gestures tend to fill in specific sectors.

Liddell and Johnson's description of "point of contact" (POC) [60] provides direct guidance for the implementation of the hand locations in virtual signing. They illustrate 20 major body locations and give detailed explanations of articulatory locations on the head, the torso, and the arm. We found that other POC features such as proximity and spatial relationships are also helpful in graphical representations.

A requirement for modeling the human body and having a hand arrive at a given location is to take into consideration the biomechanics inherent in the human body. Here, the movement of the human body is seen as the combination of motions of the body skeleton. As we mentioned in the previous chapters, the human body is modeled as a robotic manipulator with a large number of degrees of freedom (DOFs) at its joints. There are various motions at different joints. For example the wrist has two kinds of motions: (a) flexion and extension and (b) radial and ulnar deviation (side-to-side movement).

With such a large number of joints, there is no analytical solution for the problem of moving a hand to a predefined location. To simplify the location problem, the body is divided into smaller parts, for example the hand part and the arm part. Thus, the movement of a specific part becomes simplified. Software such as IKAN [12] provides an analytical and numeric solution for an anthropomorphic arm or leg. However, this is yet to be incorporate into the whole virtual figure in our sign interfacing system where there is much more to be considered. Even in the case of successful incorporation of the IKAN, there is a question of *multichoice* and *multianswer* in a sign language. For example, with the right index finger fixed on a location near the head, the various movements of the right elbow, caused by the right forearm and shoulder motions, might suggest different linguistic meanings, all of which are correct in the sign language grammar.

What we have done and are doing is the application of heuristic methods combined with the implementation of hand constraints and recording of the hand and elbow's locations. When a posture is created for the avatar, the locations of its hands and elbows are automatically calculated. These locations are part of the parametric representation of the posture and are stored in the posture database. When searching for a particular posture, we can use these locations to narrow down the search space as shown in Figure 6.2.



Right Side

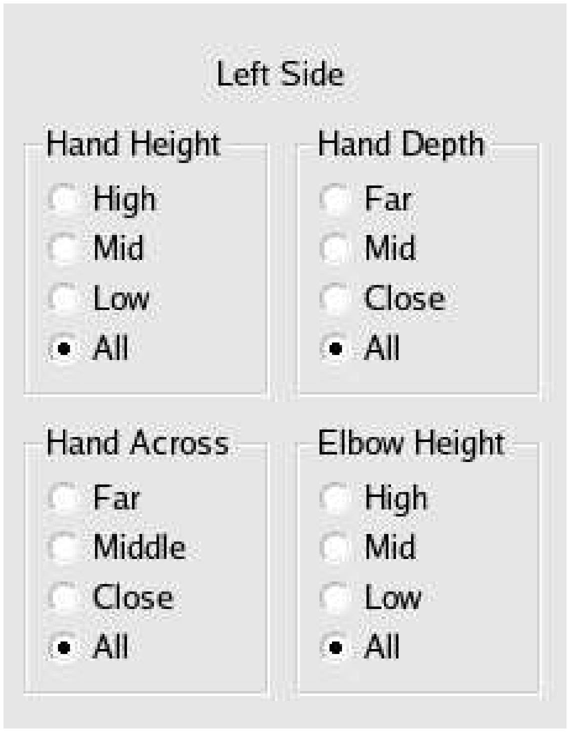
Hand Height: High Mid Low [✓ All]

Hand Depth: [✓ Far] Mid Close All

Hand Across: Far Middle Close [✓ All]

Elbow Height: High Mid Low [✓ All]

(a)



Left Side

Hand Height: High Mid Low [✓ All]

Hand Depth: Far Mid Close [✓ All]

Hand Across: Far Middle Close [✓ All]

Elbow Height: High Mid Low [✓ All]

(b)

Figure 6.2: Classification of the hand and elbow locations for limiting locations of the: (a) right hand and elbow and (b) for the left hand and elbow.

A hand's location (palm center) is classified with three types of location: *hand height, hand depth*, and *hand across.*

- *Hand height* describes how high the palm center is from the ground. Its range is divided into *High, Mid*, and *Low.*

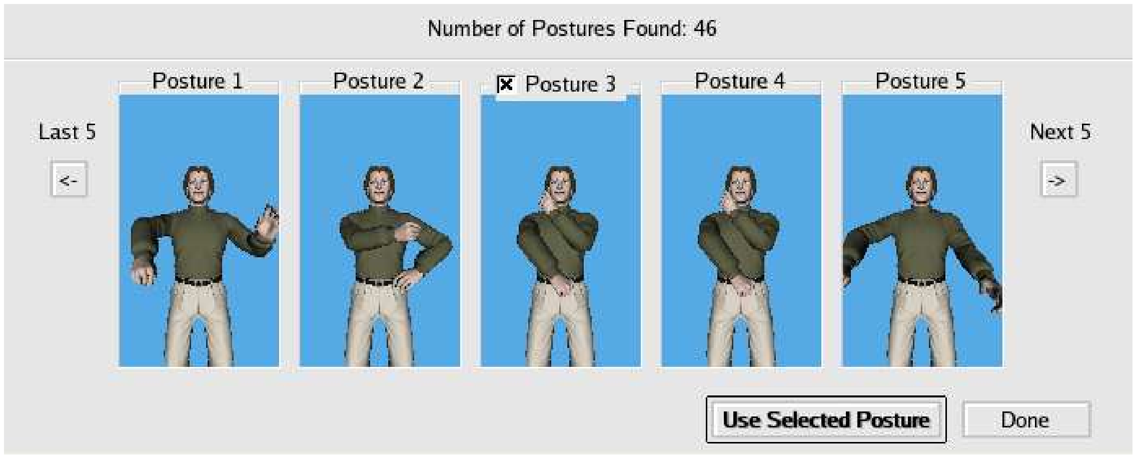
- *Hand depth* measures how far away the palm center is from the chest. Its range is divided into *Far, Mid*, and *Close.*

- *Hand Across* identifies the palm center with a horizontal right-left cross line. For example, if the right hand rests on the right side, it is marked as *Close*; when it goes across the chest to the left side, it will be on the *Far* side. This parameter is divided into three ranges: *Close, Middle*, and *Far*.

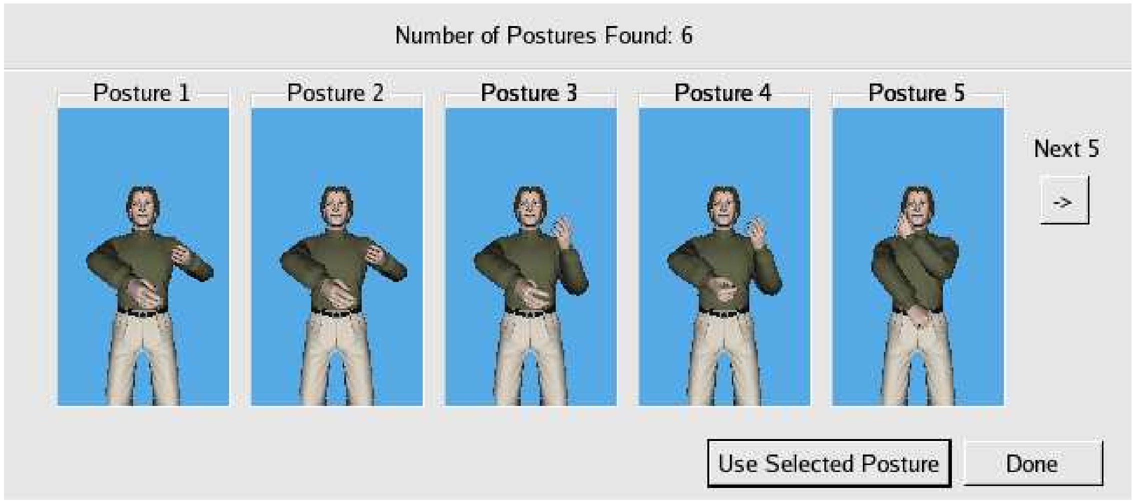
With this definition and classification, the hand's location can be represented with a set of three variables, $[across, depth, height]$, each of which has one of three different values in its range domain as defined above. There are $3 \times 3 \times 3 = 27$ different combinations to describe a hand's location. In other words, a hand's location will be in one of the 27 cubes in front of the signer, defined by three perpendicular axes in the *body coordinate system* marked with *across, depth*, and *height* in the virtual body's coordinate system with its origin at the body center.

The elbow's location is described with only one variable, *height*, which has one of three values *High, Mid*, and *Low*. Now with the consideration of locations of the avatar's two hands and two elbows, we have $3 \times 3 \times 3 \times 2 \times 3 \times 3 \times 3 \times 2 = 6,561$ different combinations of the hand and elbow's location, which means that we can divide the signing locations into $6,561$ different groups and thus greatly reduce the search space of the signing locations.

Figure 6.3 gives an example of how to use the location variables to narrow the search space for postures. We used the location classification panel in Figure 6.2 to control the search process. If we want to display or search all the postures, we set all the location variables as *All* values, and we get 46 postures (see Figure 6.3a). If we want to search postures with the right hand far away from the chest (*Far*), its height being *Low*, the right elbow having *Mid* height, the left hand in the *Mid* far away from the chest and *Mid* high, then we get only 6 postures (see Figure 6.3b).



(a)



(b)

Figure 6.3: Searching location space: (a) when all the location variables take [*All*] value, there are 46 postures; (b) when the right hand's location $[across, depth, height] = [All, Far, Low]$, the right elbow's location $[height] = [Mid]$, the left hand's location $[across, depth, height] = [All, Mid, Mid]$, and the left elbow's location $[height] = [All]$, there are only 6 postures.

6.1.3 Movements

According to Liddell and Johnson's model, signs can be composed of sequentially produced movement segments and hold segments [94]. Using this Movement-Hold model, ASL linguists describe the formation of new signs from ASL phonological and morphological parts. It is important to note that for some signs the sequence of movements and holds can be complex.

To make things easier, in our approach we disregard linguistic implications of a virtual sign. By movements, we mean any kind of movements used in signing. They can be internal movements, local movements, and movements defined by movement and hold segments. Now we use our definition of virtual gestures to define a movement segment, $mSeg$, of a sign: $mSeg = [(p_0, t_0), (p_1, t_1), ..., (p_n, t_n)]$ in which any adjacent pair of $[p_0, p_1, ..., p_n]$ will be different from each other because during movement the articulation of linguistic parts is always in a state of transition.

Following the definition of a Movement-Hold model, a hold segment is a pair of the *same* posture that occur sequentially at *different* times. For example, if the ith segment of a sign is a hold segment, $hSeg_i$, then, $hSeg_i = [(p_i, t_i), (p_{i+1}, t_{i+1})]$ where $p_i = p_{i+1}$ and $t_i \neq t_{i+1}$. For a virtual sign, $vSign$, which is composed of movement and hold segments, we have: $vSign = [mSeg_0, mSeg_1, hSeg_2, ..., mSeg_i, ..., hSeg_j, ..., mSeg_n]$ where the order of the movement and hold segments depends on the contents of the sign.

The modeling and simulation of the Movement-Hold Model in our interfacing system is quite simple: for a movement segment, construct it with postures with time factors; for a hold segment, create it with *same* posture with different times.

6.1.4 Nonmanual Signals

Nonmanual signal (NMS) or facial expression is another sign linguistic parameter. In Chapter 4, we presented the modeling of the head (including head components) and the simulation of six basic facial expressions. Although these facial expressions have been applied to the avatar of our sign language interfacing system, they have not been implemented into the system. This is because more investigation needs to be done in sign language studies on how important a role the facial expressions and head features play in the signing process and how many head and facial features should be incorporated into a virtual signing process.

A preliminary exploration on how to design a graphical user interface for applying NMS in the interfacing system will provide the design guidelines for such a task. Figure 6.4 gives an example of a prototype of an NMS creation and control panel.

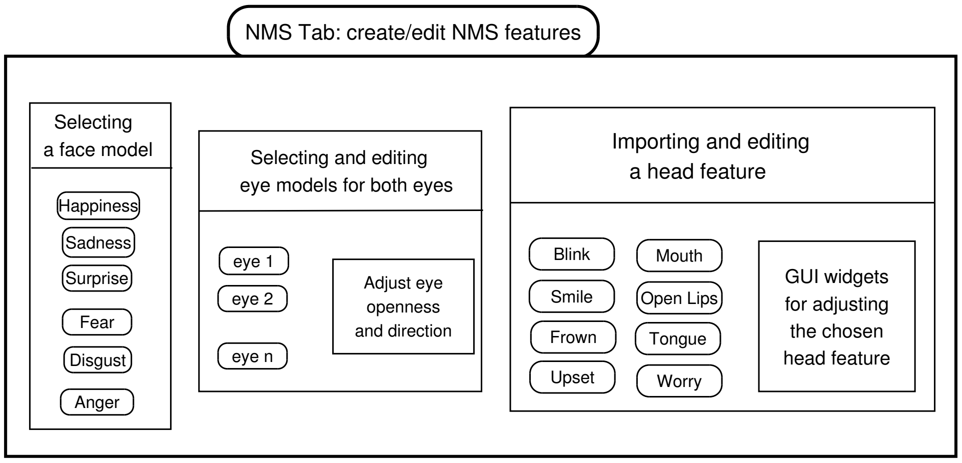

NMS Tab: create/edit NMS features
Selecting a face model
Happiness
Sadness
Surprise
Fear
Disgust
Anger
Selecting and editing eye models for both eyes
eye 1
eye 2
eye n
Adjust eye openness and direction
Importing and editing a head feature
Blink
Smile
Frown
Upset
Mouth
Open Lips
Tongue
Worry
GUI widgets for adjusting the chosen head feature

Figure 6.4: A prototype for the creation of nonmanual signals.

We consider that there are two types of NMS features: basic facial expressions and head features.

- *Basic facial expressions*: anger, disgust, fear, happiness, sadness, and surprise. Users can select one of these facial expressions, apply it to the system's avatar, and adjust the *degree* of that expression.

- *Head features*: describing the movement patterns of the mouth (lips) and eyes and the facial features that can convey some delicate feelings or emotions that are related to the six basic facial expressions such as smile, frown, upset. A head model with a neutral facial expression will be used to model and simulate these head features. Eyes are modeled according to eye features such the movements of the eyebrows, eyelids, and eyeballs' orientations. Mouth features are concerned with the movements of the lips, the tongue, and the mouth corners. In the simulation of some facial expressions, the movement patterns of the eyes and mouth might be considered.

Because of the complexity of expressing all the facial, eye, and head features, we use identification numbers (IDs) to represent these features (facialIDs, eyeIDs, and mouthIDs). Right now, we have six facialIDs for the six basic facial expressions. There is another variable for each of these facialIDs for indicating the degree of the facial expression associated with that facialID. Using feature IDs is important to the system's upgrade, because the data structures for modeling the eyes, mouth, and facial expressions might need to be expanded as new features are added into the system. At every stage of the upgrading, we can use feature IDs to retrieve the newly added features.

Figure 6.5 is a flow chart for creating, editing, and applying nonmanual signals in the system. Feature IDs are used to communicate between the avatar of the system and the NMS, eye, and facial expression databases. The facial expression databases contain data for not only the six facial expressions but also other facial expressions including mouth and head features, as described previously. This flow chart only provides a guideline to implement the sign language NMS parts and other GUIs, in addition to that shown in Figure 6.4, needed to construct and select different eye and facial expression models and to adjust the degree for each of these models.

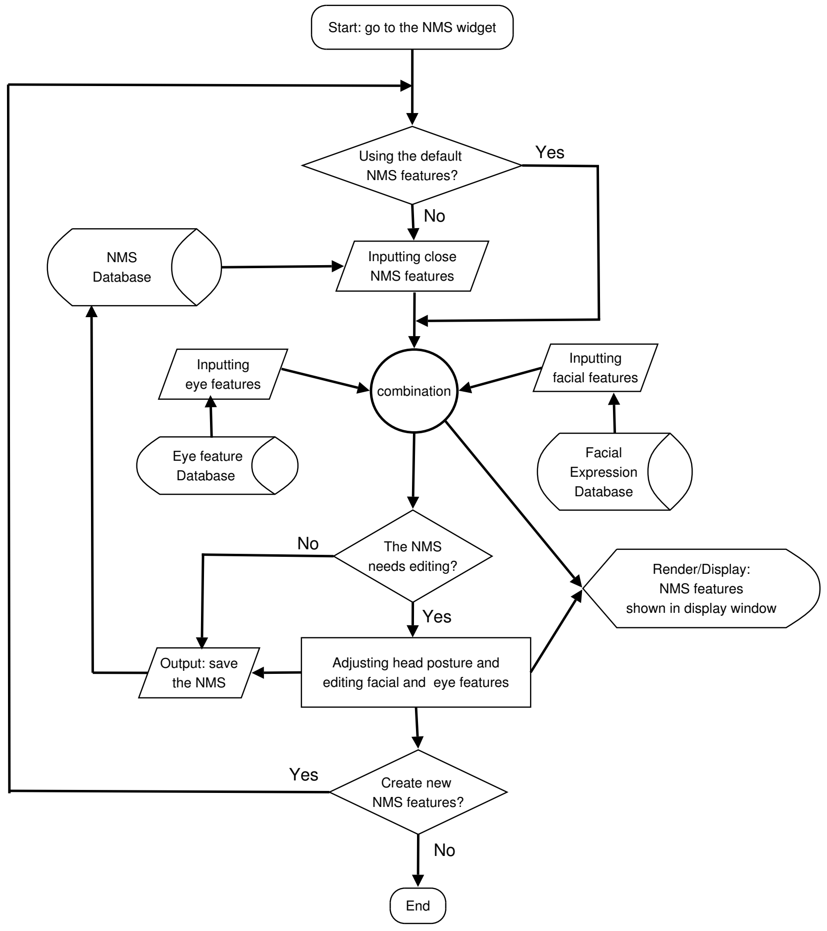

Start: go to the NMS widget
Using the default NMS features?
Yes
No
NMS Database
Inputting close NMS features
Inputting eye features
combination
Inputting facial features
Eye feature Database
Facial Expression Database
No
The NMS needs editing?
Yes
Render/Display: NMS features shown in display window
Output: save the NMS
Adjusting head posture and editing facial and eye features
Yes
Create new NMS features?
No
End

Figure 6.5: A flow chart for creating and editing nonmanual signals.

6.2 Composing Other Linguistic Parts

In the previous section, we talked about the creation of a sign language's basic linguistic parts (parameters): hand shapes and orientations, locations and gesture space, movements, and nonmanual signals. In the previous chapter, we defined a body posture, p, whose parametric (mathematical) representation is a data structure that contains a set of variables describing the body's parametric (functional) parts. Thus, all the basic linguistic parts share a common representation form: $lp = [(p_0, t_0)$, (p_1, t_1), ..., $(p_n, t_n)]$, where the contents for p_i will be different for different linguistic parts. For example, when we use lp to describe hand shapes and orientation, then only the variables in p_i that are responsible for hand shape and palm orientation will change while the other variables always have the same values. By assigning p_i and t_i with different values, we get different linguistic parts.

As in the spoken language, given some basic linguistic parts (phonemes and some morphemes), we can build from them larger linguistic parts such as morphemes, words, phrases, and even sentences, all of which can be represented as a combination of the very basic linguistic parts. Thus any of the larger linguistic parts is also an ordered list of postures: $LP = [lp_0, lp_1, ..., lp_m] = [(p_0, t_0), (p_1, t_1), ..., (p_N, t_N)]$. where lp_i indicates a basic linguistic part.

Theoretically, we can use the formula, $LP = [(p_0, t_0), (p_1, t_1), ..., (p_N, t_N)]$, to construct a sign language's words, phrases, and even sentences—that is, to create or retrieve from databases every posture, $p_0, p_1, ..., p_N$. This means that the size of LP will become unbelievably large, and this method is practically useless. So we have to use another formula: $LP = [lp_0, lp_1, ..., lp_m]$. But there is a problem in sign languages with the transition between two adjacent postures, for example, movement epenthesis, hold deletion, and assimilation in ASL. Our solution is to use both of them: $LP = [lp_0, lp_1, ..., lp_m] = [(p_0, t_0), (p_1, t_1), ..., (p_N, t_N)]$. First, we give some definitions:

- We define *wordItem* to be any of the linguistic parts, either basic or larger ones.
- Every *wordItem* has several (zero to any number in theory) *keywords* or *related words* associated with it. In the case of the creation of words and phrases for a sign language, we use *related words* for the association; in other cases, use *keywords*. But for the current version of our sign language interfacing system,

we use both of them interchangeably. Thus any of the following representation of *wordItem* will be considered correct:

- $wordItem = \{lp_i\}$: combination of any number of basic linguistic parts.
- $wordItem = \{LP_j\}$: combination of any number of larger linguistic parts.
- $wordItem = \{lp_i, LP_j\}$: combination of any number of basic and larger linguistic parts.
- $wordItem = \{mSeg_i, hSeg_j\}$: combination of any number of movement segments and hold segments in the Movement-Hold model representation.
- $wordItem = \{lp_i, LP_j, KW_m\}$ or $wordItem = \{lp_i, LP_j, RW_m\}$: combination of any number of basic and larger linguistic parts and associated *keywords* or *related words*.
- $wordItem = \{lp_i,\ [(p_k, t_k)],\ LP_j, KW_m\}$ or $wordItem = \{lp_i,\ [(p_k, t_k)],$ $LP_j, RW_m\}$: same as above, but $[(p_k, t_k)]$ indicates the inserted postures or modified postures in $\{lp_i\}$ or $\{LP_j\}$.

With so many definitions for *wordItem*, the only goal is to construct a *wordItem* in the *simplest* way under two conditions: (1) it should abide by the sign language grammar, and (2) it should assist in the construction of other linguistic parts, which is why we introduce the *keywords* or *related words*. This means that for some linguistic parts, particularly the basic linguistic parts, there will be no *keywords*, *related words*, or even “names,” only a short list of postures with time factors. For other complex linguistic parts, we can use *keywords* or *related words* to quickly search their constituents, which are then edited and combined into the new linguistic parts.

To construct a graphical user interface for creating and editing linguistic parts, we have to consider the following requirements for such an interface:

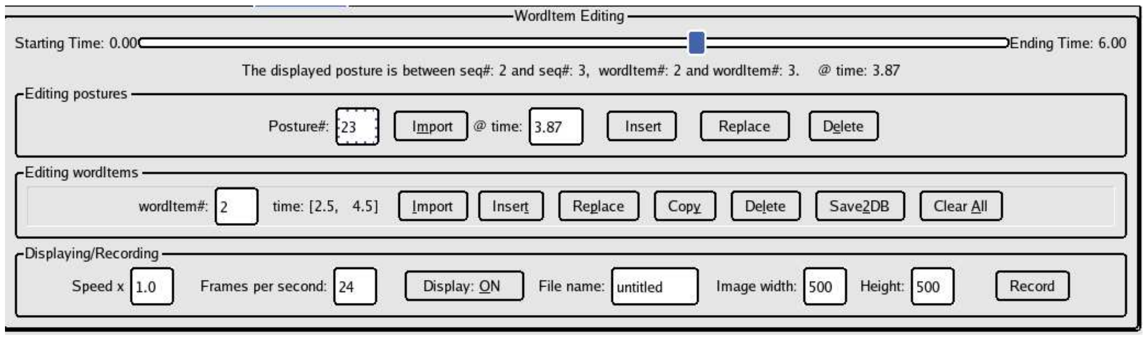
- It should be able to create and edit postures, and store to and retrieve these newly built postures.

- Based on the posture database, the interface should be able to create basic linguistic parts with time controls, edit them dynamically, store them to and retrieve them from a linguistic part database.

- It should be able to create large linguistic parts from the posture database and linguistic part database and input *keywords* or *related words* for them.

- It should provide, if possible, an editing mechanism for editing *both* the posture constituents *and* the linguistic part constituents for a linguistic part.

Figure 6.6 shows a simple implementation of the above requirements for the linguistic part creation and management interface. It looks like and works in a way similar to the *Virtual gesture creation/editing panel* shown in Figure 5.12. But with


WordItem Editing

Starting Time: 0.00 Ending Time: 6.00

The displayed posture is between seq#: 2 and seq#: 3, wordItem#: 2 and wordItem#: 3. @ time: 3.87

Editing postures

Posture#: 23 @ time: 3.87

Editing wordItems

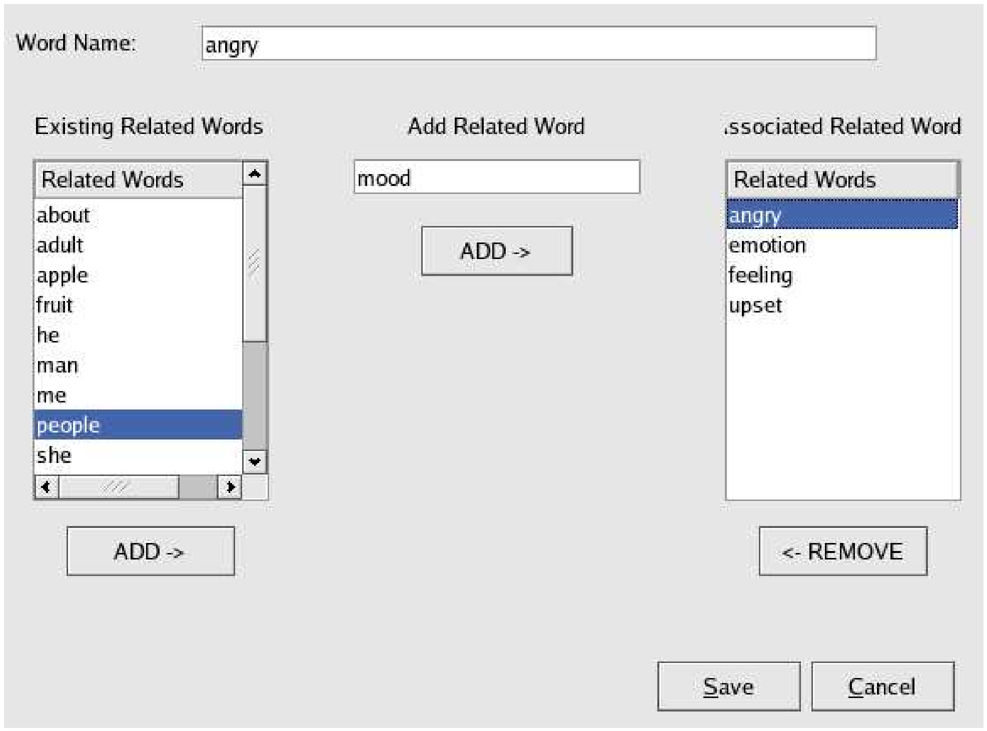
wordItem#: 2 time: [2.5, 4.5]

Displaying/Recording

Speed x 1.0 Frames per second: 24 Display: ON File name: untitled Image width: 500 Height: 500

Figure 6.6: A general control panel for creating and editing linguistic parts.

this linguistic part control panel, we can insert, delete, and edit not only the postures but also the linguistic parts. Figure 6.7 shows how to save the created linguistic parts with *Related Words*, and Figure 6.8 shows how to retrieve them with the use of *keywords*. Examples are given in Figure 6.9 and Figure 6.10, where parts of sessions of ASL words and one ASL sentence are displayed. The ASL sentence is produced in the format of *Signed English* (ASL signing but following the English syntax).


Word Name: angry

Existing Related Words

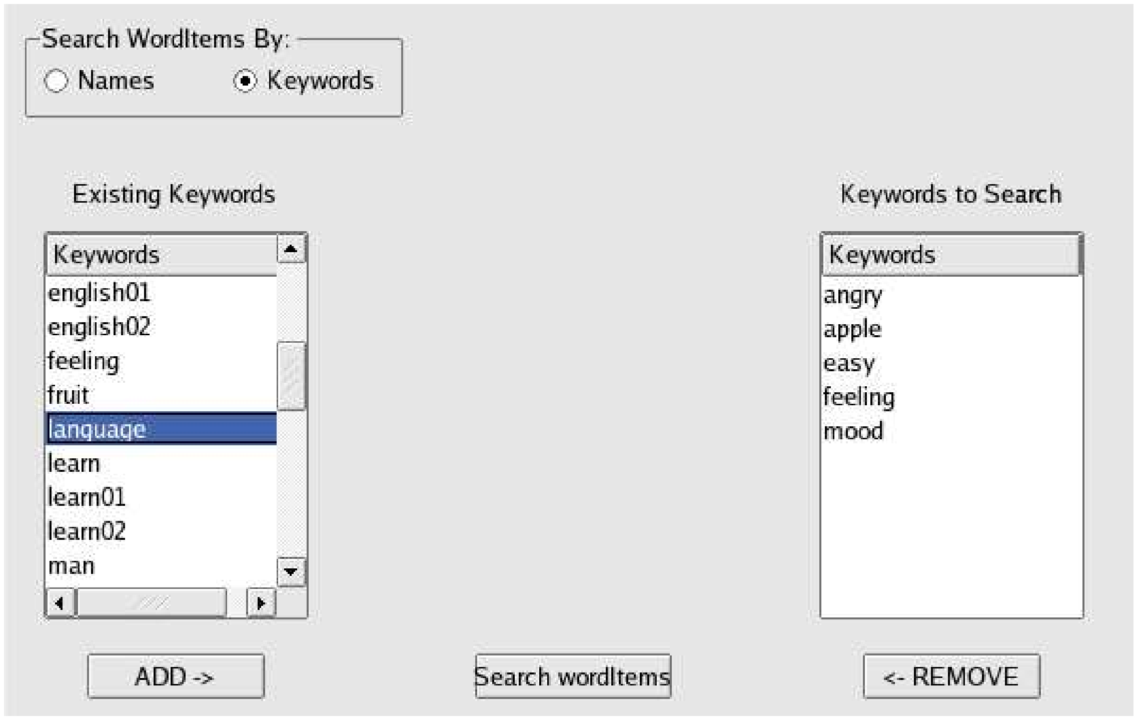
Related Words
about
adult
apple
fruit
he
man
me
people
she

Add Related Word: mood

ssociated Related Word

Related Words
angry
emotion
feeling
upset

Figure 6.7: An interface used for saving created linguistic parts.



Search WordItems By: ( ) Names [✓ Keywords]

Keywords
english01
english02
feeling
fruit
language
learn
learn01
learn02
man

Keywords
angry
apple
easy
feeling
mood

Figure 6.8: An interface used for retrieving created linguistic parts.

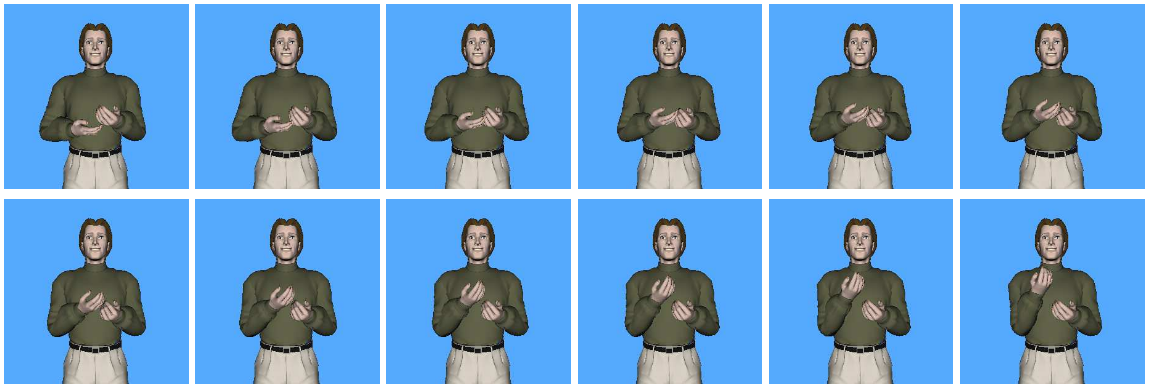


Figure 6.9: Virtual signing output of an ASL word: EASY (the session should repeat once more time).

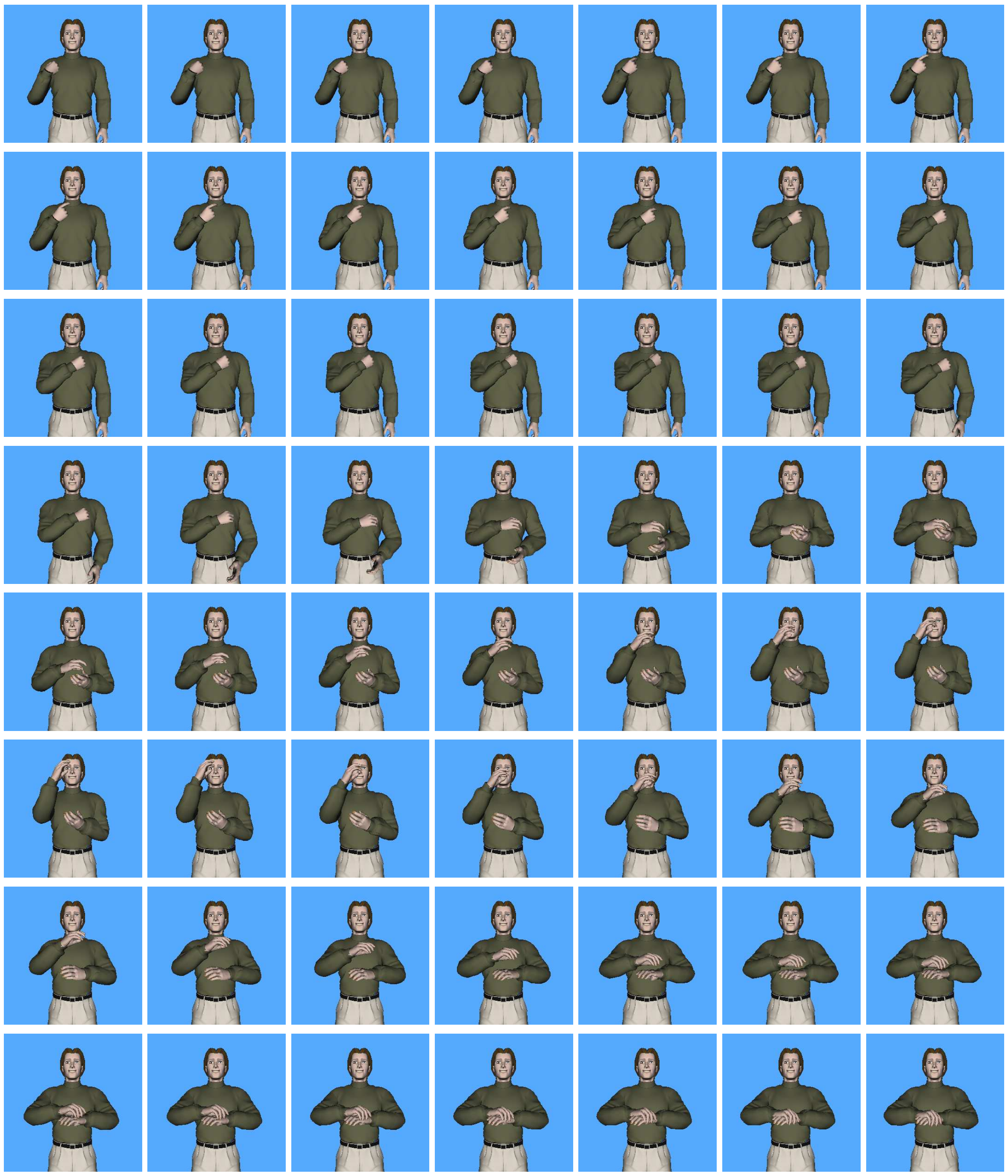


Figure 6.10: Virtual signing output (signed English) of an ASL sentence: WE LEARN ENGLISH.

6.3 "Writing" in Sign Language

It would be an ultimate goal for our sign language interfacing system to become a sign language "editor" like a text editor (such as Microsoft Word) for the spoken languages, in which users can can "write" with the system. There are three problems to consider for designing a sign language "editor": (1) selection of a sign language writing (notation) system; (2) how to retrieve (input) sign language "words" accurately in the shortest time; and (3) the transition between two adjacent "words" following the sign language syntax.

There is no standard writing system that can be used to transcribe the signing information. In fact, there may be different notation systems even in the same sign language [50]. The answer to the question of selection a notation system for the sign language interfacing system is to choose an existing one, a combination of two or more, or even create one based on the current various systems. Since our goal is to provide an all-purpose interfacing system for any sign language, this problem becomes more challenging. The assistance from and cooperation with sign language experts can help deal with this problem. After a notation system is determined for the system, there are still technical problems yet to be solved such as (1) how to design the notation icons or symbols and (2) how to combine them into the interfacing systems in editing, storing, and retrieving sign language linguistic parts.

As for the question of how to retrieve sign language "words" and "phrases," we can borrow methods such as *autocomplete* used in several Asian language text input techniques. When one clicks on a sign or types in the transcription code for a sign, the signs related to that sign (*e.g.*, with higher associated weights or sharing the first transcription coding symbols) will be displayed on the screen, each of which is an animation sequence accompanied by a number or notation symbols, rendered in an easy-to-understand style in a small screen area. One can easily click on the desired sign or type in its representative number or notation symbols. Figure 6.11 gives a simple example.

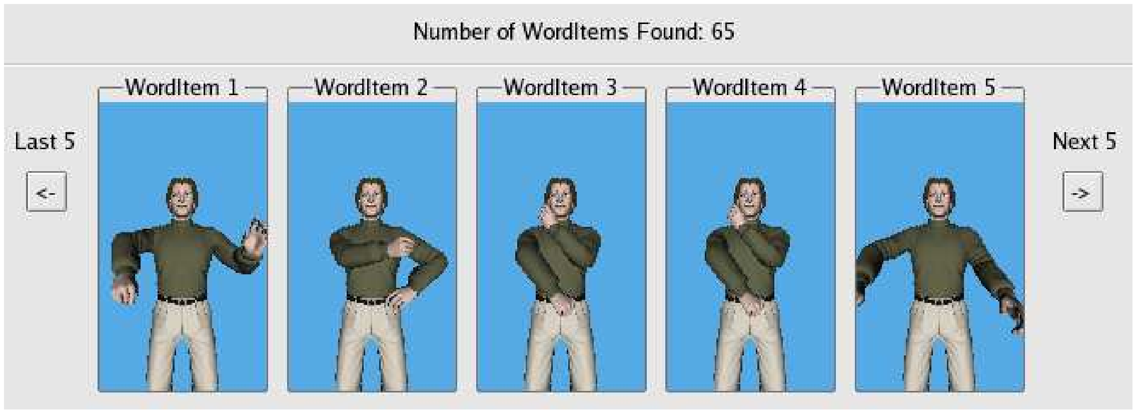


Figure 6.11: An example of how to quickly retrieve a sign language "word": these display windows display virtual signs (animation sessions) for different but related (associated) "words"; the word "WordItem" above the display window is supposed to be notation symbols of a sign language for the sign below it. The user will have three input methods to choose from: (1) clicking on the virtual sign in a display window, (2) typing in the notation symbols, and (3) typing the number (1, 2, ...) above the virtual sign.

An important requirement for this *autocomplete* solution is that a sign should be rendered in different styles, corresponding to various situations. A life-like virtual animation can be used as the end production of a sequence of signs, with outline and wireframe renderings providing very fast (yet understandable) representations for a large number of signs in many small-sized display windows at the same time so that the user can recognize and select signs.

In a text editor for spoken languages, letters, words, and phrases are sequentially juxtaposed, but in a sign language there is a transitional process between two signing parts in which the two parts exert influence over each other, following the syntax rules of a sign language. This means that postures (including their corresponding time factors) on the border of two adjacent signs have to be changed. For example, there are four typical variations in a phonological process in ASL: movement epenthesis, hold deletion, metathesis, and assimilation [94]. In graphical implementations, this presents a movement-control-over-time design problem. Thus, the interfacing system should offer a mechanism that can quickly adjust (edit) the motions of avatar parts during a limited time interval. A promising solution might be to build a database with a large number of "phrases" in the forms such as *noun and verb* pairs, *compounds*, and *classifier predicates*. In addition, a carefully designed retrieval graphical user interface together the application of the *autocomplete* method will greatly assist in the retrieval of these "phrases."

Chapter 7

Conclusions and Future Work

This dissertation presents work in computer graphics, human-computer interaction, and user interface design. We introduce a sign language interfacing system that can serve as a working platform for sign language users in creating and managing sign language linguistic parts. Design guidelines and specifications for the system have been proposed and explained in detail. The implementation of the proposed functions for the system has produced exploratory and interesting results to support the prospect and plausibility of an interfacing system for sign languages.

First, a life-like human avatar was constructed following the structure of a human male body. This virtual human figure was then divided into a set of functional parts according to the movement patterns of the human body parts. There are three major groups of body parts: the head together with the eyes for modeling facial expressions, the hand for modeling a variety of hand shapes, and upper body parts, including the shoulders, upper arms and forearms, for body gesture simulation.

Various motions of the different human body parts have been investigated and extracted as a set of parametric variables. Specific data structures have been defined which correspond to the different sets of parametric variables for the description of the movements of different body parts, such as hand configurations, upper body movements, and facial expressions. Thus, by assigning appropriate values to the parametric variables, we can simulate human gestures.

Finally, with the implementation of user-friendly graphical interfaces to wrap up all the operations on the avatar and the specifically-designed gesture databases to manage the virtual gestures, we can construct any desired virtual gestures and associate them with sign language linguistic parts.

The proposed system is only a framework for a complete sign language interfacing system, and there are many areas that need to be improved:

- Incorporation of more hand constraints into the interfacing system will improve the simulation of natural hand configurations.
- Modeling and simulation of natural movements at the shoulder is always a challenging topic. Implementing the most recent results in computer graphics on modeling a shoulder will produce smoother and more genuine real-life shoulder movements.
- The investigation of using inverse kinematics in the avatar and testing the results might be helpful in the simulation of human gestures.
- There is a usability consideration for the interfacing system. Invite sign language experts to test the software and get feedbacks and comments. Set up an interdisciplinary team to improve the interfacing system.
- The selection and/or design of a sign language notation system, and the reduction the avatar data size so that it can be rendered very quickly in the many small display windows for a sign "editor" will make this sign language interfacing system a practical and workable software.

Bibliography

[1] Athena Review Image Archive [online, cited May 10, 2006]. http://www.athenapub.com/sraphand.htm.

[2] Control Gestures [online, cited May 10, 2006]. http://www.gvu.gatech.edu/ccg/publications/iswc2000-pendant/img6.png.

[3] DePaul ASL Synthesizer [online, cited May 10, 2006]. http://asl.cs.depaul.edu/.

[4] eSign: Essential Sign Language Information on Government Networks [online, cited May 10, 2006]. http://www.visicast.sys.uea.ac.uk/eSIGN/.

[5] Eye Movements in Natural Tasks [online, cited May 10, 2006]. http://www.cis.rit.edu/people/faculty/pelz.

[6] Facial animation and modeling [online, cited May 10, 2006]. http://www.mpi-sb.mpg.de/resources/FAM/.

[7] HamNoSys [online, cited May 10, 2006]. http://www.sign-lang.uni-hamburg.de/Projects/HamNoSys.html.

[8] Hand – Posteroanterior (PA) View, Labelled [online, cited May 10, 2006]. http://www.rad.washington.edu/RadAnat/HandPALabelled.html.

[9] Hand Model [online, cited May 10, 2006]. http://www.handmodel.ca/.

[10] Hand Opposition Example [online, cited July 3, 2003]. http://www.dpt.state.va.us/comcur/commcurr/2000/mar/hand.gif.

[11] Human Performance Evaluation of Interaction Tecnologies [cited July 3, 2003]. http://hci.stanford.edu/~craig/hci/lectures/zhai/1/sld012.htm.

[12] IKAN: Inverse kinematics using an analytical method [online, cited May 10, 2006]. http://hms.upenn.edu/software/ik/.

[13] KDevelop. http://www.kdevelop.org/.

[14] Laban writer [online, cited May 10, 2006]. http://www.dance.ohio-state.edu/labanwriter/.

[15] Maya software [online]. http://www.mayahtt.com/.

[16] MySQL. http://www.mysql.com/.

[17] OpenGL. http://www.opengl.org/.

[18] Perl. http://www.perl.com/.

[19] Qt Programming. http://www.trolltech.com/.

[20] The Coin Source [online]. http://www.coin3d.org.

[21] UPENN HMS Center. http://hms.upenn.edu/ [cited May 10, 2006].

[22] Vcom3d. http://vcom3d.com/ [cited May 10, 2006].

[23] Virtual reality lab. http://vrlab.epfl.ch/ [cited May 10, 2006].

[24] *A Dictionary of American Sign Language on Linguistic Principles.* Linstsok Press, 1965.

[25] *A Dictionary of American Sign Language on Linguistic Principles.* Linstsok Press, 1976.

[26] Johanny Accot and Shumin Zhai. Beyond Fitts' Law: models for trajectory-based HCI tasks. In *ACM Proc. of CHI*, pages 395–302, 1997.

[27] Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. Construction and animation of anatomically based human hand models. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 89–109, San Diego, California, July 2003.

[28] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 612–619, San Antonio, TX, July 2002.

[29] Amaury Aubel and Daniel Thalmann. Realistic deformation of human body shapes [online]. 2000 [cited May 10, 2006]. http://vrlab.epfl.ch/Publications/publications_index.html.

[30] Norman Badler, Jan Allbeck, Liwei Zhao, and Meeran Byun. Representing and parameterizing agent behaviors. In *in Proc. Computer Animation, IEEE Computer Society*, pages 133–143, Geneva, Switzerland, 2002. http://www.cis.upenn.edu/~badler/paperlist.html [cited May 10, 2006].

[31] Norman I. Badler and Jan M. Allbeck. Towards behavioral consistency in animated agents, 2001. http://www.cis.upenn.edu/~badler/paperlist.html [cited May 10, 2006].

[32] Norman I. Badler, Cary B. Phillips, and Bonnie L. Webber. *Simulating Humans: Computer Graphics, Animation, and Control.* Oxform University Press, 1993. http://www.cis.upenn.edu/~badler/book/book.html [cited May 10, 2006].

[33] Carliss Y. Baldwin and Kim B. Clark. *Design Rules: The Power of Modularity.* MIT Press, 2000.

[34] Sir Charles Bell. *The Hand: Mechanism and Vital endowments, as Evincing Design.* Harper & Brothers, New York, 1855.

[35] Jules Bloomenthal. Medial-based vertex deformation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 147–151, San Antonio, TX, July 2002.

[36] R. Boulic et al. The HUMANOID environment of multiple deformable human characters, 1995. `http://vrlab.epfl.ch/Publications/publications_index.html` [cited May 10, 2006].

[37] Paul W. Brand. *Clinical Mechanics of the Hand.* The C. V. Mosby Company, 1985.

[38] Steve Capell, Seth Greeen, Brian Curless Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 586–593, San Antonio, TX, July 2002.

[39] Edmund Y. S. Chao, Kai-Nan An, William P. Cooney III, and Ronald L Linscheid. *Biomechanics of the Hand: A Basic Research Study.* World Scientific Publishing Co. Pte. Ltd., Singapore, 1989.

[40] Chin-Seng Chua, Haiyin Guan, and Yeong-Khing Ho. Model-based 3D hand posture estimation from a single-2D image. *Image and Vision computing*, 20(3):191–202, March 2002.

[41] Michae C. Corballis. *From Hand to Mouth: The Origins of Language.* Princeton University Press, Princeton, 2002.

[42] George ElKoura and Karan Singh. Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 110–119, San Diego, California, July 2003.

[43] Karen Emmorey. *Language, cognition, and the Brain: Insights from Sign Language Research.* Erlbaum, Mahwah, N. J., 2002.

[44] Deborah I. Fels, Jan Richards, Jim Hardman, Sima Soudian, and Charles Silverman. American sign language of the web. In *CHI 2004, Late Breaking Results Paper*, Vienna, Austria, April 2004.

[45] Nancy Frishberg, Serena Corazza, Linda Day, Sherman Wilcox, and Rolf Schulmeister. Sign language interfaces. In *Proceedings of the ACM/INTERCHI'93 conference on Human Factors in Computing Systems*, pages 194–197, 1993.

[46] Jonathan Gratch et al. Creating interactive virtual humans: some assembly required. `http://www.cis.upenn.edu/~badler/paperlist.html` [cited May 10, 2006].

[47] Thomas T. Hewett and et al. Curricula for human-computer interaction. `http://www1.acm.org/sigs/sigchi/cdg/` [cited May 10, 2006]. ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group.

[48] Gregory Hickok, Ursula Bellugi, and Edward S. Klima. SIGN language in the BRAIN. *Scientific America*, 12(1):46–53, August 2002.

[49] Matthew Paul Huenerfauth. Design approaches for developing user-interfaces accessible to illiterate users. In *American Association of Artificial Intelligence (AAAI 2002) Conference, Intelligent and Situation-Aware Media and Presentations Workshop.*

[50] Matthew Paul Huenerfauth. A survey and critique ofamerican sign language natural language generation and machine translation systems, technical report. Computer and Information Sciences, Univeristy of Pennsylvania, September 2003.

[51] Doug L. James and Dinesh K. Pai. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 582–585, San Antonio, TX, July 2002.

[52] Kolja Kähler, Jörg Haber, Hitoshi Yamauchi, and Hans-Peter Seidel. Head shop: Generating animated head models with anatomical structure. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 55–63, San Antonio, TX, July 2002.

[53] E. Klima and U. Bellugi. *The Signs of Language.* Harvard University Press, Cambridge, MA, 1988.

[54] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: Real time large deformation character skinning in hardware. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 153–159, San Antonio, TX, July 2002.

[55] Sumedha Kshirsagar and Nadia Magnenat-Thalmann. A multilayer personality model. In *Proceedings of the 2nd International Symposium on Smart Graphics*, pages 107–115, Hawthorne, NY, 2002.

[56] James J. Kuch and Thomas S. Huang. Vision-based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *IEEE Proceedings of Fifth International Conference on Computer Vision*, pages 666–671, June 1995.

[57] Jintae Lee and Tosiyasu L. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, pages 77–86, September 1995.

[58] J. P. lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *ACM SIGGRAPH Proceedings*, pages 163–172, New Orleans, LA, 2000.

[59] Scott. K. Liddell, editor. *Grammar, Gesture, and Meaning in American Sign Language.* Gallaudt University, Washington DC, 2003.

[60] Scott. K. Liddell and R. E. Johnson. American sign language: The ponological base. *Sign Language Studies*, Fall(64):195–227, 1989.

[61] John Lin, Ying Wu, and Thomas S. Huang. Modeling the constraints of human hand motion. In *in Proc. 5th Annual Federated Laboratory Symposium (ARL2001)*, pages 105–110, Maryland, April 2001.

[62] Jeffrey Loomis, Howard Poizner, Ursula Bellugi, Alynn Blakenore, and John Hollerbach. Computer graphic modeling of american sign language. *Computer Graphics*, 17(3):105–114, July 1983.

[63] I. Scott MacKenzie. Movement time prediction in human-computer interfaces. http://www.yorku.ca/mack/GI92.html, 1996.

[64] Walter Maurel. *3D Modeling of the Human Upper Limb including the biomechanics of Joints, Muscles and Soft Tissues.* PhD thesis, Laboratoire d'Infographie — Ecole Polytechnique Federale de Lausanne, 1998.

[65] Walter Maurel and Daniel Thalmann. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics*, 24(2):203–218, 2000.

[66] David McNeil. *Hand and Mind: What Gestures Reveal About Though.* The University of Chicago Press, Chicago, 1992.

[67] David McNeil, editor. *Language and Gesture.* Cambridge University Press, 2000.

[68] Carol Meidle, Stan Sclaroff, and Vassilis Athitsos. SignStream: A tool for linguistic and computer vision research on visual-gestural language data. *Behavior Research Methods, Instruments, & Computers*, 33(3):311–320, 2001.

[69] Keith Miller. User-centered interface design. `http://www.uis.edu/~miller /se/tutorial.html`.

[70] Laurent Moccozet and Nadia Magnenat Thalmann. Multilevel deformation model applied to hand simulation for virtual actors, 1998. `http://miralabwww.unige.ch/newMIRA/ARTICLES/VSMM97B.pdf` [cited June, 2002].

[71] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. In *SIGGRAPH 2003*, San Diego, CA.

[72] Louise A. Montoya, Reginald Egnatovitch, Elizabeth Eckhardt, Marjorie Goldstein, Richard A. Goldstein, and Annie G. Steinberg. Translation chanllenges and strategies: The ASL translation of a computer-based, psychiatric diagnostic interview. *Sign Language Studies*, 4(4):314–344, 2004 Summer.

[73] Brad A. Myers. A brief history of human computer interaction technology. `http://www.victoriapoint.com/hci_history.htm`. ACM interactions. Vol. 5, no. 2, March, 1998. pp. 44-54.

[74] John Napier. *Hands.* Pantheon Books, New York, 1980.

[75] Jean-Christonphe Nebel and Alexander Sibiryakov. Range flow from stereo-temporal matching: Application to skinning. In *IASTED International Conference on Visualization, Imaging, and Image Processing*, Malaga, Spain, 2002.

[76] Luciana Porcher Nedel and Daniel Thalmann. Modeling and deformation of the human body using an anatomically-based approach, 1998. `http://vrlab.epfl.ch/Publications/publications_index.html` [cited May 10, 2006].

[77] Carol Neidle. SignStream: A database tool for research on visual-gusture language, report no. 10, ASL linguistic research project, August 2000. `http://www.bu.edu/asllrp/` [cited May 10, 2006].

[78] American Academy of Orthopaedic Surgeons. *Joint Motion: Method of Measuring and Recording*. Churchill Livingstone, New York, 1988.

[79] David Lorge parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, December 1972.

[80] Hans Rijkema and Michael Girard. Computer animation of knowledge-based human grasping. *Computer Graphics*, 25(4):339–348, July 1991.

[81] Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In *ACM SIGGRAPH Proceedings*, pages 163–172, 1997.

[82] Andrew Sears. Hci education: Where is it headed? `http://www.victoriapoint.com/hcied.htm`.

[83] Rajeev Sharma, Vladimir I. Pavlovic, and Thomas S. Huang. Toward multimodal human-computer interface. `http://www.ifp.uiuc.edu/IDFL/papers/paperssharma.html#sharmaab18` [cited May 10, 2006].

[84] Maryann Simmons, Jane Wilhelms, and Allen Van Gelder. Model-based reconstruction for creature animation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 139–146, San Antonio, TX, July 2002.

[85] Karan Sing and Eugene Fiume. Wires: a geometric deformation technique. In *ACM SIGGRAPH Proceedings*, 1998.

[86] Peter-Pike J. Sloan, III Charles F. Rose, and Michael F. Cohen. Shape by example. In *ACM Symposium Interactive 3D Graphics*, 2001.

[87] William C. Stokow. Visible verbs become spoken. *Sign Language Studies*, 5(2):152–169, 2005.

[88] David Joel Sturman. *Whole-hand Input*. PhD thesis, Massachusetts Institute of Technology, Febrary 1992. `http://xenia.media.mit.edu/~djs/thesis.ftp.html` [cited May 10, 2006].

[89] Kevin J. Sullivan, Willam G. Griswold, Yuanfang Cai, and Ben Hallen. The structure and value of modularity in software design. In *Proceedings of the 8th European Software Engineering Conference*, pages 99–108, Vienna, Austria. ACM SIGSOFT, ACM Press, New York, USA.

[90] Daniel Thalmann and Jean-Sébastien Monzani. behavioural animation of virtual humans: what kind of laws and rules?, 2002. `http://vrlab.epfl.ch/Publications/publications_index.html` [cited May 10, 2006].

[91] Thomas and et al. Speech and vision integration for display control. `http://www.ifp.uiuc.edu/IDFL/research/ressharma.html`.

[92] Thomas and et al. Speech and vision integration for display control. `http://www.ifp.uiuc.edu/IDFL/results/resultshuang.html` [cited May 10, 2006].

[93] Sutton Valerie. `http://www.signwriting.org/` [cited May 10, 2006].

[94] Clayton Valli and Ceil Lucas. *Linguistics of American Sign Language: An Introduction*. Gallaudet University Press, third edition, 2000.

[95] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 129–138, San Antonio, TX, July 2002.

[96] Sherman Wilcox. The multimedia dictionary of american sign language: Learing lessons about language, technology, and business. *Sign Language Studies*, 3(4):379–392, 2003 Summer.

[97] Sherman Wilcox, Doug Wood, Dennis Cokely, and William C. Stokoe. Multimedia dictionary of american sign language. In *Proceedings of the First Annual ACM Conference on Assistive Technologies*, pages 9–16, Marina Del Rey, CA, 1994.

[98] Jane Wilhelms. Animals with anatomy. *IEEE Computer Graphics and Applications*, pages 22–30, May-June 1997.

[99] Ying Wu and Thomas S. Huang. Hand modeling, analysis, and recognition for vision-based human computer interaction. *IEEE Signal Processing Magazine*, 18(3):51–60, May 2001.

[100] Yu Zhang, Edmond C. Prakash, and Eric Sung. A new physical model with multilayer architecture for facial expression animation using dynamic adaptive mesh. *IEEE Transactions on Visualization and Computer Graphics*, pages 339–352, May 2004.